Universidad Zaragoza
1542

Trabajo Fin de Grado

Alineación automática y escalable de emociones georreferenciadas a cartografía de referencia y visualización flexible de las mismas

Automatic and scalable alignment of georeferenced emotions to reference cartography and flexible visualization of them

Autor

Sergio Costa Moreno

Director

F. Javier Zarazaga Soria

Escuela de Ingeniería y Arquitectura
2019/2020

ALINEACIÓN AUTOMÁTICA Y ESCALABLE DE EMOCIONES GEORREFERENCIADAS A CARTOGRAFÍA DE REFERENCIA Y VISUALIZACIÓN FLEXIBLE DE LAS MISMAS

RESUMEN

Una emoción es una alteración del ánimo intensa y pasajera, agradable o penosa, que suele asociarse a reacciones psicofisiológicas que representan modos de adaptación a ciertos estímulos del individuo cuando percibe un objeto, persona, lugar, suceso o recuerdo importante. Es decir, los lugares, y por extensión lo que los caracterizan, influyen en las emociones. Cobra sentido, por tanto, tomar en consideración los elementos que constituyen los espacios físicos para ver cómo influyen en las personas a través de las emociones que provocan.

Este trabajo fin de grado forma parte de una línea de investigación en relación con las emociones, el entorno; y cómo este puede actuar sobre ellas con el fin de desarrollar posibles sistemas que puedan operar sobre los sentimientos que una persona puede tener en un determinado lugar y así influir en su comportamiento. Más concretamente, la motivación de este proyecto se basa en la vinculación de las emociones recogidas a entornos específicos, en este caso calles de núcleos urbanos. La integración de un gran volumen de estos datos es lo que servirá como base de estudio en trabajos futuros, de los que se podrán extraer conclusiones en función de parámetros como la edad, el género, el tramo horario o la estación del año.

Los problemas que se abordan en este TFG son la generación de una base cartográfica que sirva de referencia para asociar emociones georreferenciadas con su entorno, la vinculación entre dichas emociones y segmentos de calle, y, dado que la base cartográfica representa la realidad física y ésta cambia con el tiempo, el problema de desactualización de los datos cartográficos descargados. La resolución de estos problemas se sustenta en una aplicación de gestión que permite la descarga bajo demanda (establecida por las coordenadas geográficas de las emociones) de los datos de OpenStreetMap, la vinculación de las emociones a tramos de calle (resolviendo posibles conflictos que se plantean), y la programación de los procesos de refresco que acompañan al mantenimiento de las versiones locales de la base de datos de OpenStreetMap para garantizar las actualizaciones. El sistema se completa con una pequeña aplicación web de demostración que posibilita dar visibilidad a todos los trabajos de vinculación llevados a cabo.

Este proyecto se integra en los trabajos de una línea de colaboración puesta en marcha por tres grupos de investigación de la Universidad de Zaragoza (Affective Lab, DISCO e IAAA) para el desarrollo de conocimiento y tecnología para la identificación, caracterización, y explotación mediante sistemas de información de las emociones que las personas sienten en diferentes lugares, y desarrollando diferentes actividades.

DECLARACIÓN DE AUTORÍA

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. SERGIO COSTA MORENO ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO (Título del Trabajo)

Alineación e integración de emociones georreferenciadas y presentación de resultados por colores en mapa de calles

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24/06/2020

Fdo: SERGIO COSTA MORENO

ÍNDICE

AGRADECIMIENTOS

Me gustaría agradecer a mis padres la oportunidad de poder completar esta titulación sin preocupaciones externas, contando siempre con su apoyo.

A los profesores que forma parte del proyecto, y en especial a Javier Zarazaga, por su gran ayuda y consejo en la realización de éste.

Y desde luego, a mis compañeros de grado por acompañarme en las dificultades y alegrías de estos últimos cuatro años.

INTRODUCCIÓN

1.1 Contexto del Trabajo

Según la RAE, una emoción es una alteración del ánimo intensa y pasajera, agradable o penosa, que va acompañada de cierta conmoción somática[1]. En la Wikipedia se extiende este concepto vinculándolo a reacciones psicofisiológicas que representan modos de adaptación a **ciertos estímulos** del individuo cuando percibe un objeto, persona, **lugar**, suceso o recuerdo importante[2]. Es decir, los lugares, y por extensión lo que los caracterizan, influyen en las emociones. Cobra sentido, por tanto, tomar en consideración los elementos que constituyen los espacios físicos para ver cómo influyen en las personas a través de las emociones que provocan.

Este proyecto se integra en los trabajos de una línea de colaboración puesta en marcha por tres grupos de investigación de la Universidad de Zaragoza para el desarrollo de conocimiento y tecnología para la identificación, caracterización, y explotación mediante sistemas de información de las emociones que las personas sienten en diferentes lugares, y desarrollando diferentes actividades. Estos grupos de investigación son Affective Lab (http://giga.cps.unizar.es/affectivelab/), DISCO (http://webdiis.unizar.es/DISCO/) e IAAA (http://iaaa.unizar.es). Los primeros resultados de esta colaboración de se vinculan al proyecto DJRunning (https://djrunning.es) cuyo objetivo es el desarrollo de conocimiento y tecnología orientados a la motivación y mejora del rendimiento de los corredores de media maratón y maratón por medio de la música.

1.2 Contexto Tecnológico

Dado que los requisitos esenciales del sistema no obligan a la utilización de ninguna tecnología en particular, y con la excusa de profundizar en competencias menos desarrolladas a lo largo del grado, se ha un propuesto un uso poco homogéneo de tecnologías.

El núcleo del sistema se implementa con Spring Boot[3], entorno de trabajo para el lenguaje Java[4] ampliamente utilizado por su gran número de facilidades a la hora del desarrollo. Este proporciona soporte para bases de datos relacionales y NoSQL[5] mediante Spring Data[6], esencial en este proyecto para configurar de forma sencilla accesos a los diferentes tipos de persistencia que se utilizan: PostgreSQL[7], MongoDB[8] y MySQL[9]. Además, implementa un fácil manejo de dependencias basado en anotaciones que junto a las configuraciones básicas de Spring hacen que el despliegue de aplicaciones sea simple y rápido.

---

[1] https://dle.rae.es/emoción
[2] https://es.wikipedia.org/wiki/Emoción
[3] https://spring.io/projects/spring-boot
[4] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)
[5] https://es.wikipedia.org/wiki/NoSQL
[6] https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference
[7] https://es.wikipedia.org/wiki/PostgreSQL
[8] https://es.wikipedia.org/wiki/MongoDB
[9] https://es.wikipedia.org/wiki/MySQL

Angular[10] es un entorno de trabajo para desarrollar la capa de presentación de aplicaciones web, comúnmente se utiliza dentro del estándar MEAN[11] (MongoDB, Express[12], Angular, Node.js[13]) por la fácil integración con el resto de las tecnologías. En el marco de este proyecto, se ha utilizado tanto en la aplicación para gestión por parte de administradores, como en una pequeña aplicación para la presentación de los resultados obtenidos. Esta segunda implementa el estándar mencionado, la parte de Angular presenta los elementos que interactúan con el usuario, Node.js junto con la biblioteca Express conforman la parte lógica, MongoDB es la tecnología detrás de la capa de persistencia.

Quartz[14] se ha utilizado para la programación periódica de descarga de emociones y actualización de datos geográficos. Es un entorno de programación de tareas periódicas, también llamado "scheduling", de código abierto que provee funcionalidad avanzada para la calendarización de tareas en Java.

Puntualmente se ha hecho uso de scripts en Python[15] para generar datos falsos de forma aleatoria o para transformar ficheros JSON en GeoJSON[16], ambas situaciones orientadas a pruebas con datos geográficos.

1.3 Motivación y problema que se aborda

Como se ha visto anteriormente, las emociones, y su variabilidad, pueden provenir de los lugares por los que nos movemos o nos hemos movido. Sin embargo, tal y como se puede ver por el enunciado de la frase, estamos hablando de hacer análisis a tiempo pasado. Si tratamos de desarrollar sistemas predictivos que operen sobre las emociones que un usuario siente, o va a sentir, (como es el caso del proyecto DJRunning mencionado) básicamente tenemos dos opciones:

1. Esperar a que nuestro usuario pase una primera vez por un lugar y capturamos la emoción que siente en el mismo. A partir de ese momento podríamos asumir que siempre sentirá la misma emoción en ese lugar. ¿Pero es esto realmente así? La emoción puede variar atendiendo a factores tales como la hora del día o la época del año. Eso sin entrar en otras consideraciones de carácter más aleatorio como son si llueve o no, temperatura, etc. Además, estaríamos obligando a nuestro usuario a pasar siempre una primera vez por todos los sitios.
2. Identificar patrones de emociones asociados a lugares en función de la captura de las mismas de muchos potenciales usuarios. De este modo, las emociones sentidas por unos usuarios pueden ser la base para inferir las de otros. Esto permite, además, que podamos prever las emociones de un usuario en un lugar por donde nunca ha pasado.

Este proyecto forma parte de una línea de trabajo que trata de dotarse de unas bases de conocimiento sobre las emociones que las personas sienten en diferentes lugares. Con estas

[10] https://es.wikipedia.org/wiki/AngularJS
[11] https://es.wikipedia.org/wiki/MEAN
[12] https://en.wikipedia.org/wiki/Express.js
[13] https://en.wikipedia.org/wiki/Node.js
[14] http://www.quartz-scheduler.org/
[15] https://es.wikipedia.org/wiki/Python
[16] https://es.wikipedia.org/wiki/GeoJSON

bases se podrán llevar a cabo desarrollos de sistemas capaces de operar sobre los sentimientos que una persona puede tener en un determinado lugar de cara poder influir en su comportamiento. En el marco de otro trabajo final de grado en marcha se está desarrollando una aplicación móvil que servirá de herramienta para la recolección de las emociones que diferentes lugares inspiran a los usuarios. A partir de estas emociones que se vinculan a coordenadas geográficas, surge la necesidad de extender esta percepción a un espacio físico que resulte lógico desde el punto de vista de la realidad en la que un usuario se está moviendo. Concretamente, la motivación de este proyecto está en la necesidad de poder vincular las emociones recogidas a elementos cartográficos específicos, en este caso, a calles de entornos urbanos. Dado que una calle puede ser un elemento cartográfico muy grande y heterogéneo (veamos por ejemplo la Avenida Diagonal de Barcelona que tiene 10,2 Km[17]), se ha decidido trabajar usando elementos más pequeños y que suponen los tramos de calle que van entre cruce y cruce de calles. A partir de aquí, y en posteriores trabajos, cuando un usuario se encuentre en el mismo segmento de calle que otros que ya han mostrado sus emociones, será posible llevar a cabo inferencias tomando como punto de partida a las mismas.

Los problemas que se abordan en este proyecto son:

- La integración de los datos recogidos por la mencionada aplicación móvil en una única base de datos centralizada. De cara a facilitar el despliegue de la aplicación en entornos geográficos concretos, esta aplicación se está construyendo para que cuente con una base de datos propia en cada despliegue. Es necesario solventar el problema de integrar los sucesivos despliegues en un entorno de gestión que permita ir periódicamente a buscar datos de emociones recopiladas.
- Vinculación de todas las emociones recopiladas a segmentos de calle concretos. Para ello será necesario abordar varios subproblemas:
    - Contar con una base cartográfica de referencia que provea de los segmentos de calle a los que vincular las emociones. La opción elegida es OpenStreetMap[18] por ser un proyecto que ofrece cobertura mundial y acceso libre a los datos. El tamaño que tiene la propia base de datos de OpenStreetMap hace que operacionalmente no resulte lógico tener toda ella de partida (por el alto coste que tendría su descarga para no tener garantizado un alto % de uso de la misma, en secciones sucesivas se dan detalles), por lo que uno de los problemas a tratar es la descarga progresiva en función de las necesidades que se vayan teniendo.
    - Vincular las posiciones de una emoción a un segmento dado. Es necesario contar con un procedimiento que, ante la posición de una emoción, determine cuál es el segmento al que se vincula (cabe mencionar que los sistemas de seguimiento por satélite que son los que proveen de las coordenadas geográficas de las emociones tienen errores de precisión[19]).
- Disponer de un sistema de gestión que automatice procesos de mantenimiento de datos. La base cartográfica de referencia (OpenStreetMap) es un elemento vivo que crece en precisión (incorporando cada vez más información), a la vez que refleja la evolución de los entornos reales que describe (cambios urbanísticos como nuevas calles o modificación de existentes). Es por ello por lo que periódicamente habrá que rehacer

[17] https://es.wikipedia.org/wiki/Avenida_Diagonal_(Barcelona)
[18] https://www.openstreetmap.org
[19] https://www.redeweb.com/articulos/el-camino-hacia-el-posicionamiento-de-alta-precision-en-el-mercado-masivo/

las operaciones de vinculación de emociones a segmentos de calle sobre las versiones más nuevas de los datos.

A lo largo de la memoria de este trabajo final de grado se va a mostrar cómo han sido abordados estos problemas y qué soluciones se les han dado.

Este proyecto, es el primero realizado que pretende continuar su ciclo de vida una vez finalizado, acercándose más a lo que sería un contexto laboral. Por este simple motivo, se ha prestado especial atención a todo el proceso de desarrollo, queriendo dejar patente que todo lo aprendido en los últimos cuatro años se ha interiorizado suficientemente como para abordar un trabajo de tal magnitud. Además, el implementar un sistema tan diferente a las tareas realizadas en las asignaturas del grado, ha supuesto desde el principio un incentivo.

1.4 Alcance, objetivos y limitaciones

El objetivo básico de este TFG es disponer de un sistema que permita la vinculación de emociones georreferenciadas a una base cartográfica de escala mundial y que va evolucionando con el tiempo. Tal y como se ha mencionado, esta evolución de la base cartográfica necesita de automatismos para que las propias vinculaciones de las emociones se ajusten a la misma.

El alcance del proyecto viene establecido por la resolución de los problemas indicados anteriormente: integración de los datos recogidos por la aplicación móvil en sus múltiples despliegues, vinculación de estos datos a una base cartográfica, y actualización automática de los datos acorde con la evolución de la base cartográfica. Este alcance se sustenta en el desarrollo de dos aplicaciones web: una orientada a la gestión interna del sistema y que da respuesta a los mencionados problemas; y otra cuyo objetivo es dar visibilidad a los procesos de gestión mediante la presentación de los resultados obtenidos sobre un entorno interactivo.

La aplicación de gestión comprende la administración de los almacenes externos de los cuales se extraen las emociones, y la configuración de las tareas automatizadas: recolección de emociones y actualización de datos geográficos.

En la aplicación de presentación de resultados, se incluyen segmentos de calle coloreados sobre un mapa. Dichos segmentos aparecerán respetando los límites impuestos por un filtro, que atiende a parámetros horarios, de género, de edad y de tipo de usuario.

El sistema que funciona por debajo tiene como objetivo extraer las emociones georreferenciadas de almacenes externos, comprobar si los datos geográficos pertinentes han sido ya descargados, descargarlos y filtrarlos en caso negativo y asociar la emoción al segmento de calle más cercano. Al mismo tiempo, este se encarga de actualizar periódicamente la parte geográfica por si se producen modificaciones en la vía pública.

Los límites del proyecto vienen establecidos por la aplicación móvil de captura de emociones (que como se ha mencionado es objeto de otro TFG), así como el análisis y extracción de conclusiones de la información recolectada (que queda como trabajo de continuación de este TFG).

1.5 Herramientas de trabajo

La realización del proyecto se apoya tanto en herramientas relacionadas con el propio grado (editor de código fuente, etc.) como en herramientas de uso más común (Google Drive, Skype).

Herramientas de uso no técnico:

- Google Drive para la compartición de ficheros entre tutor y alumno.
- Google Meet para comentar brevemente los avances semanales.
- Skype (v. 14.56.102.0) y Discord (v. 0.0.306) para el planteamiento de problemas o dudas al tutor de forma excepcional.
- Trello para la gestión de tareas pendientes.

Herramientas de uso técnico:

- Visual Studio Code (v. 1.46.1) como entorno de desarrollo.
- Overpass-turbo para la formación de Open Street Map.
- QGIS Desktop (v. 3.10.2) para las pruebas de asociación de segmentos y emociones.
- MongoDB Compass (v. 1.20.5.0) para gestionar la base de datos MongoDB[20].
- pgAdmin 4 (v. 4.19) para gestionar la base de datos PostgreSQL[21].
- MySQL Command Line Client (v. 8.0.20) para las pruebas con emociones generadas de forma aleatoria.
- Postman (v. 7.26.1) para simular las peticiones *http* que realizarían los diferentes módulos sin necesidad de que estén construidos.
- GitHub Desktop (v. 2.5.2) para el control de versiones.

Adicionalmente, se detallan a continuación las características de la máquina y el entorno de trabajo con el que se ha desarrollado el proyecto ya que a lo largo de esta memoria se incluyen referencias a tiempos de descarga y procesado de información que se han ejecutado con esta máquina y este entorno.

- Equipo de desarrollo: HP ENVY Laptop 13-ad0xx con procesador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 2904 MHz y memoria RAM de 8 GB.
- Red de comunicaciones: conexión WI-FI a 126,84 MB/s de bajada y 28,39 MB/s de subida.

1.6 Esquema general de la memoria del proyecto

La memoria de este trabajo fin de grado comienza con una introducción a la línea de investigación, a las tecnologías con las que se ha trabajado, a los problemas que se abordan y al objetivo de estos. Posteriormente, se explican de forma detallada los requisitos del sistema y cómo se han implementado. Esta explicación se encuentra distribuida en subapartados desde el 2.1 hasta el 2.5.

[20] https://es.wikipedia.org/wiki/MongoDB
[21] https://es.wikipedia.org/wiki/PostgreSQL

El primero de los anexos, Anexo I, da una visión general de qué es OpenStreetMap, cómo funciona, y qué acciones han sido necesarias para poder consumir información de él.

Se adjuntan varios anexos que exponen con mayor detalle el funcionamiento interno de los componentes de la estructura. En el Anexo II se explica cómo se lleva a cabo la descarga de datos geográficos y su refresco. En el Anexo III se detalla la vinculación entre segmentos y emociones, cómo se realizaron las primeras pruebas y cómo queda finalmente el algoritmo para resolver situaciones de conflicto.

El Anexo IV expone el problema existente al filtrar un gran volumen de datos de cara a la aplicación de presentación de resultados, y cómo se ha solucionado precalculando ciertas consultas.

Con el Anexo V se da una visión del manejo de ambas aplicaciones desarrolladas, se indica desde cada pantalla qué puede hacer el usuario y qué acciones conlleva.

TRABAJO DESARROLLADO

2.1 Requisitos del sistema

Esta sección introduce los requisitos que han sido considerados para el sistema de información que se ha desarrollado en este TFG.

Requisitos funcionales del sistema global:

S1	El sistema debe poder descargar emociones de almacenes externos cada cierto tiempo definido.
S2	El sistema debe poder descargar datos geográficos.
S3	El sistema debe poder asociar emociones a su segmento más cercano.
S4	El sistema debe poder actualizar datos geográficos cada cierto tiempo definido.

Requisitos funcionales para un usuario externo:

U1	Un usuario debe poder visualizar un mapa con segmentos de calle coloreados según la emoción más recurrente en cada tramo.
U2	Un usuario debe poder filtrar las emociones representadas en el mapa según tramo horario, mes, estación, género, edad y tipo de usuario.

Requisitos funcionales para un administrador:

A1	Un administrador debe poder iniciar y cerrar sesión en la aplicación.
A2	Un administrador debe poder visualizar las direcciones donde se alojan almacenes externos de información con emociones.
A3	Un administrador debe poder comprobar que la ruta de un almacén externo es accesible.
A4	Un administrador debe poder añadir un almacén externo.
A5	Un administrador debe poder modificar un almacén externo.
A6	Un administrador debe poder eliminar un almacén externo.

Requisitos no funcionales:

NF1	Los colores que representan cada emoción deben ser: nervioso-color, asqueado-color, neutro-color, contento-color, relajado-color.
NF2	Una emoción debe tener asociada al entrar al sistema dos marcas de tiempo, dos pares de coordenadas, la edad y género del usuario, el tipo de usuario y un valor entre 1 y 5 (incluidos) que represente la emoción.
NF3	La base cartográfica de referencia debe ser OpenStreetMap.
NF4	La descarga de datos geográficos se debe realizar por pequeños lotes de tamaño fijo que en su conjunto conforman todo el planeta.
NF5	El tamaño de los lotes de descarga se debe definir en función de pruebas de carga.

Diccionario de datos:

- Tipo de usuario: representa ciudadano o turista.
- Almacén externo: representa un nombre, una uri y un usuario y contraseña.
- Tarea: representa descarga, alineamiento e integración de emociones o; actualización de datos geográficos ya descargados.

2.2 Arquitectura software del sistema

Desde un punto de vista de la arquitectura del sistema, el objetivo es desarrollar un sistema que, admitiendo algún cambio en su configuración, funcione de forma automática descargando emociones de almacenes externos y alineándolas con segmentos de calle. Es por ello por lo que se ve necesaria la creación de ciertos componentes con un propósito definido y un componente que haga de coordinador.

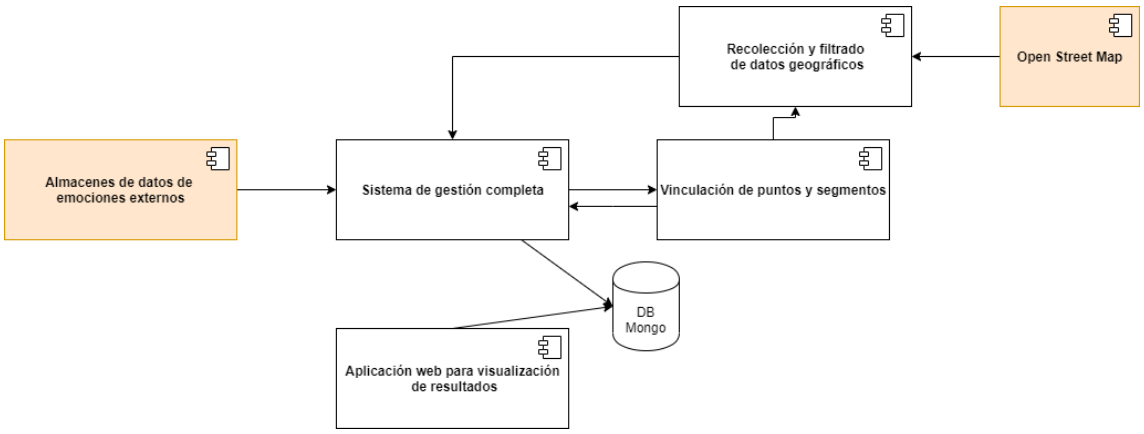

Figura 1. Concepto inicial de la arquitectura del sistema

Se plantea un componente dedicado exclusivamente a la recolección de datos de OpenStreetMap. Este es llamado por la vinculación de puntos y segmentos si en la base de datos

se tiene descargada una zona geográfica requerida, y por el sistema de gestión para que datos geográficos descargados pasado un tiempo sean actualizados.

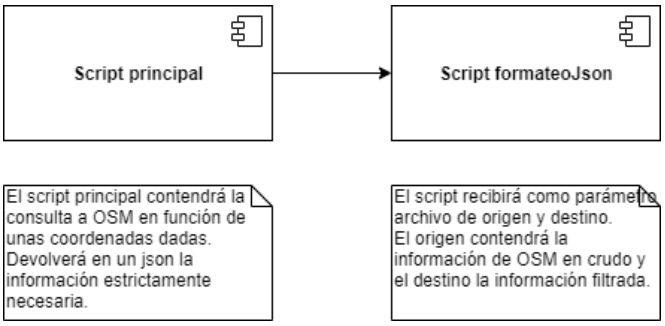

Figura 2. Concepto inicial de la recolección y filtrado de datos geográficos

La vinculación de puntos y segmentos tiene como único cometido asociar emociones georreferenciadas por un par de coordenadas, con el segmento de calle más cercano.

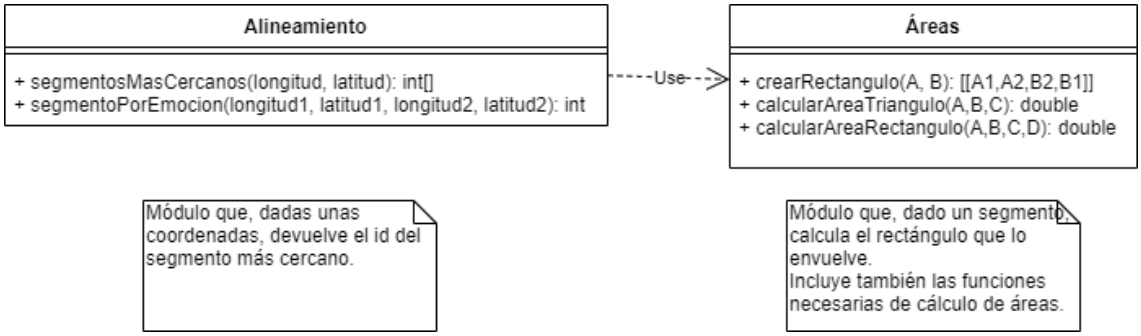

Figura 3. Concepto inicial de la vinculación de puntos y segmentos

Los almacenes externos son los proveedores de las emociones que se recogen periódicamente, y tanto la periodicidad de dicha recolección como la de la actualización de datos geográficos, son configurables a través de una aplicación web. Esta además permite la gestión de los almacenes externos.

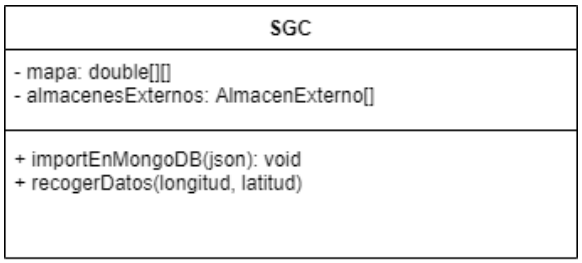

Figura 4. Concepto inicial de sistema de gestión completa

Por otro lado, se encuentra la aplicación web para la visualización de resultados. Esta consume de la propia base de datos en la que se guardan las emociones ya alineadas. Su único cometido es presentar información filtrando según ciertos parámetros.

2.3 Diseño del sistema

Diagramas de módulos

La aplicación de gestión se divide en dos grandes paquetes, uno con la lógica interna del sistema (*spring-boot-quartz*) y otro para la capa de presentación (*frontend*).

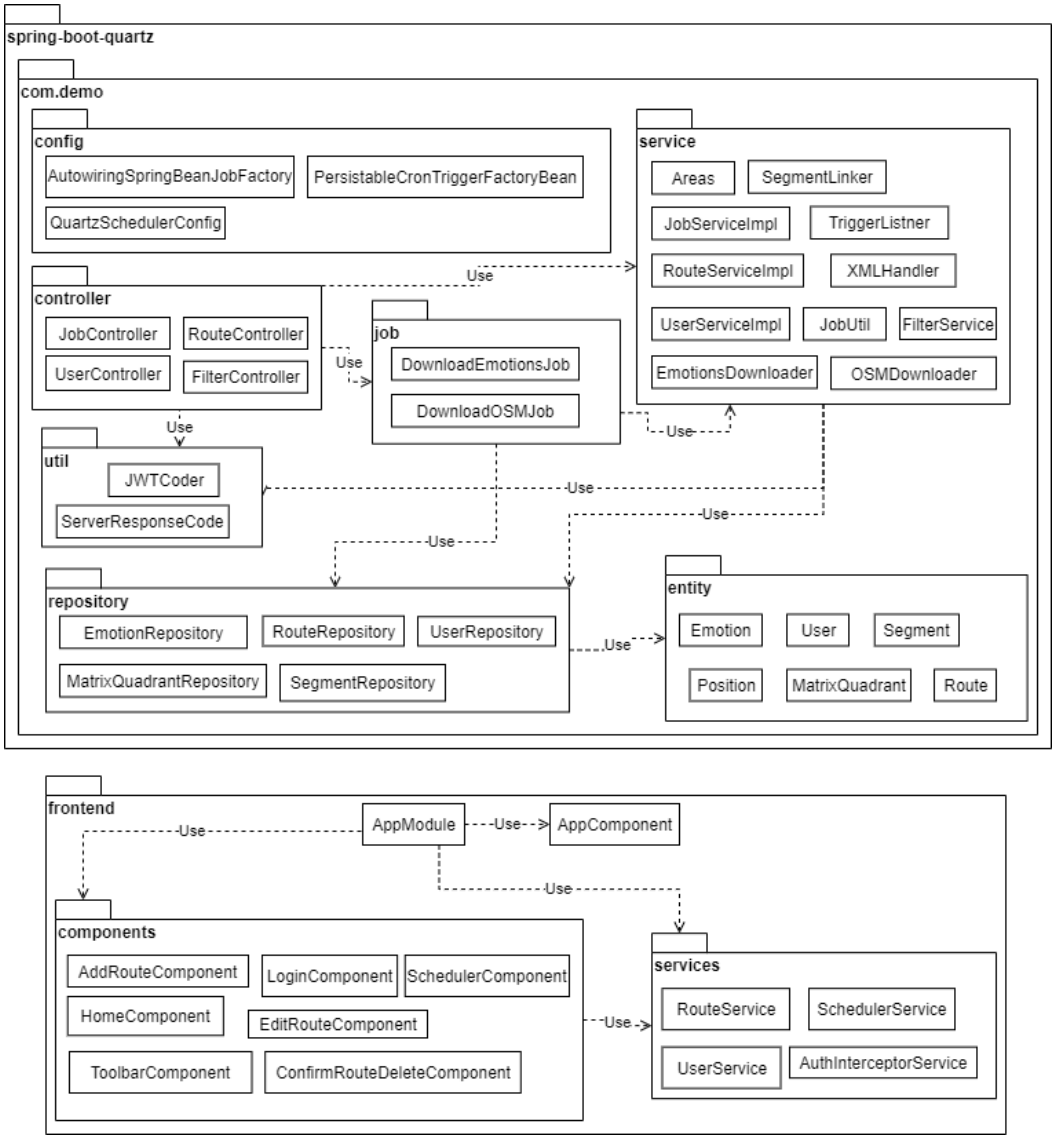

Figura 5. Diagrama de módulos de la aplicación de gestión

El primero de ellos está compuesto por paquetes más pequeños según la función que desempeñan. En *entity* se encuentra la representación del dominio mediante emociones, usuarios, cuadrantes, segmentos y rutas. A su vez, *repository* contiene los diferentes repositorios que encapsulan la persistencia de dichas entidades. El paquete *service* agrupa las clases en las que se apoyan las tareas de recolección de emociones y actualización de datos geográficos (*job*)

y los controladores (*controller*). Dichos controladores exponen una interfaz REST[22] de la que consume la capa de presentación.

El paquete *frontend* incluye en *components* las diferentes vistas y elementos que el administrador ve en la aplicación web. Estas vistas son: una página para iniciar sesión (*LoginComponent*), la página principal donde se ven los almacenes externos con emociones (*HomeComponent*), una página para añadir (*AddRouteComponent*) y otra para editar (*EditRouteComponent*) la información referida a los almacenes y una última para configurar las tareas automatizadas (*SchedulerComponent*). *ConfirmRouteDeleteComponent* presenta al usuario la confirmación de que desea eliminar el acceso a un almacén externo. En el paquete *services* se incluyen las peticiones a la lógica de la aplicación (paquete *spring-boot-quartz*). *AppModule* y *AppComponent* son clases de configuración de las descritas previamente.

Diagramas de componentes y conectores

Al analizar el sistema en tiempo de ejecución se observa como el sistema de alineación e integración de datos y la aplicación web de visualización de resultados acceden a la misma base de datos, aunque es solo el primero el que modifica su información.

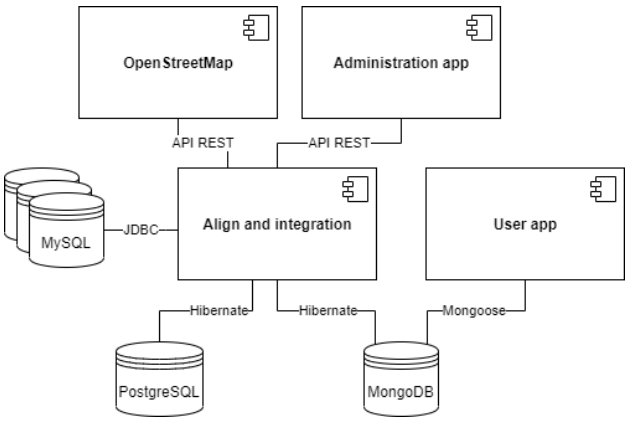

Figura 6. Diagrama de componentes y conectores general

El sistema central se encuentra conectado a OpenStreetMap para la descarga de la información geográfica, dicha información es accesible mediante una API REST y se guarda en la base de datos MongoDB junto con las emociones ya alineadas y la ruta a uno o varios almacenes MySQL. En MySQL se encuentran las emociones todavía no procesadas y se accede gracias a JDBC[23]. PostgreSQL se encarga de la persistencia de las tareas programadas que definen el comportamiento del sistema, es accesible por medio de Hibernate[24], igual que MongoDB. La aplicación de administración tiene como cometido la programación de las posibles tareas y la gestión de las rutas a los almacenes MySQL.

La descarga de los datos geográficos y la vinculación de segmentos y emociones se describen respectivamente en Anexo II y Anexo III.

[22] https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional
[23] https://es.wikipedia.org/wiki/Java_Database_Connectivity
[24] https://es.wikipedia.org/wiki/Hibernate

Modelo de datos

En la base de datos MongoDB se almacenan rutas a almacenes externos, casillas de la matriz mundial (consultar Anexo I), segmentos de calle, usuarios y emociones ya asignadas a un segmento.

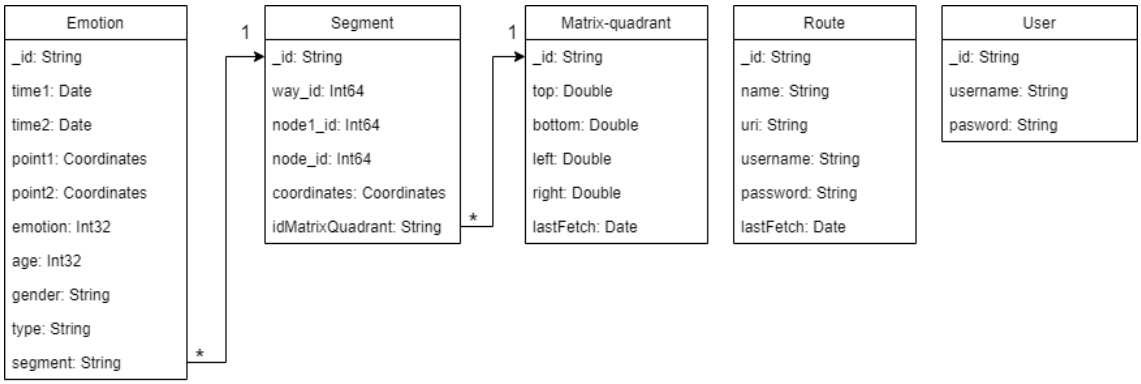

Figura 7. Modelo de datos en MongoDB

Las emociones (Emotion) están compuestas de dos momentos en el tiempo y dos posiciones (point1 y point2). Este diseño de dos puntos y dos momentos en el tiempo (consecutivos de tan apenas unos segundos configurables en el despliegue de la aplicación) ha sido legado de la aplicación móvil de captura de emociones. La idea es que cuando se le pide al usuario que etiquete un lugar se toman dos posiciones en dos momentos cercanos en el tiempo con un doble motivo. El primero, y principal, es facilitar el proceso de vinculado de la emoción a un segmento de calle ya que, al tener dos puntos muy cercanos en el tiempo, es más fácil discernir, tal y como se detalla en el Anexo III. Por otro lado, esta doble captura de puntos abre la puerta a un futuro análisis de las emociones basado en la velocidad de desplazamiento ya que sería posible estimarla "restando" los dos puntos. Las emociones se completan con:

- El valor propio de la emoción que es un entero entre 1 y 5 con la siguiente semántica asociada:

1	Nervioso	😰
2	Asqueado	🤢
3	Neutro	😐
4	Contento	😃
5	Relajado	😌

- La edad que es un número entero.
- El género que puede tomar tres valores: M.- Masculino; F.- Femenino; y O.- Otro.
- El tipo que corresponde con la diferenciación de si el usuario es un turista o no.
- El segmento al cual el proceso de emparejamiento ha vinculado la emoción.

El identificador de cada segmento se construye concatenando el identificador de la vía, el id de un nodo y el id del otro nodo de la siguiente forma: <*way_id*>_<*node1_id*>_<*node2_id*>. Para evitar conflictos *node1_id* es siempre el mínimo entre *node1_id* y *node2_id*. Un segmento puede estar asociado a varias emociones y pertenece a una sola casilla (*Matrix-quadrant*).

Cada casilla almacena los bordes que la definen y la fecha de su última descarga, esta fecha es la que determina si ha quedado desactualizada y es necesario volver a descargar sus segmentos de OpenStreetMap.

Las rutas a las bases de datos externas incluyen además de la dirección y sus credenciales, la fecha de la última descarga de emociones; con este dato se pueden filtrar las emociones y descargar únicamente las posteriores a dicha fecha para evitar duplicados.

En MySQL están guardadas las emociones antes de ser procesadas. Corresponden directamente con los datos que son capturados por la aplicación móvil. Como se puede observar, los campos son los mismos que para Emotion con la excepción del soporte a la implementación de la relación con el segmento.

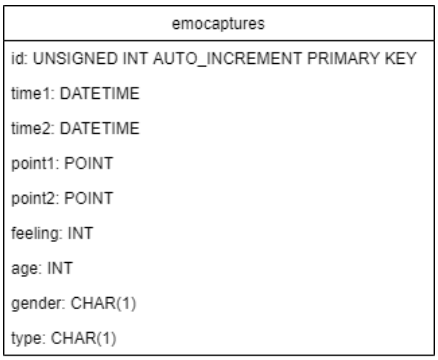

Figura 8. Modelo de datos en MySQL

Finalmente, en PostgreSQL se encuentra la información referida a las tareas programadas.

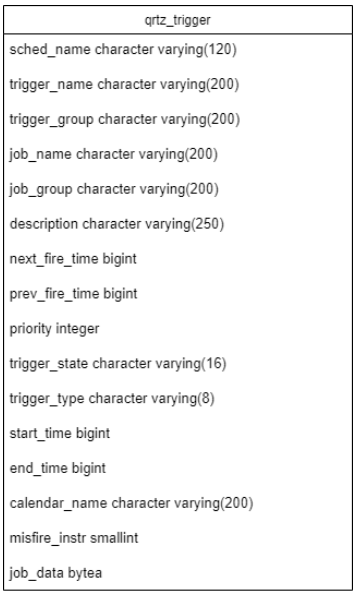

Figura 9. Modelo de datos en PostgreSQL

El tipado de los datos se corresponde con los definidos en cada tipo de almacenamiento.

2.4 Dimensión del trabajo realizado

En una primera propuesta del proyecto se planteó dedicar el mes de febrero a la formación en las tecnologías que posteriormente se desarrollarían, marzo para toma de contacto con los datos geográficos,y desde marzo hasta junio desarrollo del sistema por completo. El desarrollo del sistema comprendía la vinculación de segmentos y emociones con datos experimentales, el desarrollo de la aplicación web para muestreo de resultados y la automatización de los procesos de ampliación del sistema.

Este esquema no ha sido modificado en ningún momento, los tiempos de implementación de cada pieza del sistema se han ajustado de forma casi perfecta a la planificación inicial. Se han dedicado un total de 215 horas. Debido a los problemas que se detallan posteriormente, en algún momento se ha estimado que el tiempo de dedicación debía aumentar, pero en ningún momento se ha valorado posponer la entrega.

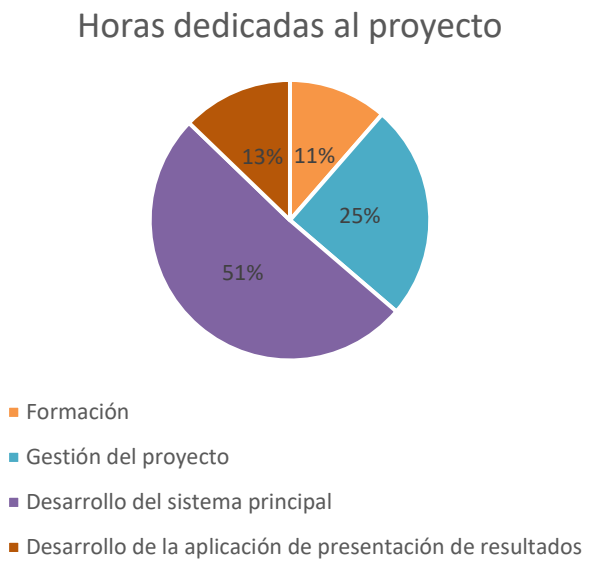

Figura 10. Horas dedicadas al proyecto

Una vez dado por finalizado el desarrollo del trabajo fin de grado se hace un recuento de 36553 líneas de código. Se reparten de forma mayoritaria en 29586 en ficheros JSON, 3096 en ficheros Java y 1532 en ficheros TypeScript[25].

[25] https://es.wikipedia.org/wiki/TypeScript

```
total files : 191
total code lines : 18968
total comment lines : 494
total blank lines : 1031

   statistics
|      extension|     total code|  total comment|    total blank|percent|
-------------------------------------------------------------------------
|               |            162|             23|             19|   0.85|
|            .py|             37|             12|             10|   0.20|
|          .json|          13429|              0|             27|     71|
|            .js|             52|              9|              4|   0.27|
|            .md|             22|              7|             17|   0.12|
|           .xml|            116|             23|             26|   0.61|
|          .html|            311|             15|             10|    1.6|
|           .css|            129|              1|             32|   0.68|
|            .ts|           1165|            105|            222|    6.1|
|         .prefs|             20|              3|              0|   0.11|
|    .properties|             68|              2|              6|   0.36|
|           .sql|            330|              8|             36|    1.7|
|          .java|           3096|            286|            622|     16|
|           .lst|             31|              0|              0|   0.16|
-------------------------------------------------------------------------
```

Figura 11. Líneas de código de la aplicación de administración

```
total files : 61
total code lines : 17585
total comment lines : 178
total blank lines : 238

   statistics
|      extension|     total code|  total comment|    total blank|percent|
-------------------------------------------------------------------------
|               |             96|             47|             29|   0.55|
|          .json|          16157|              0|             27|     92|
|            .js|            300|             21|             35|    1.7|
|            .md|             14|              5|             13|  0.080|
|          .jade|             16|              0|              2|  0.091|
|            .ts|            367|             80|             66|    2.1|
|           .css|            596|             25|             58|    3.4|
|          .html|             38|              0|              8|   0.22|
|       .geojson|              1|              0|              0| 0.0057|
-------------------------------------------------------------------------
```

Figura 12. Líneas de código de la aplicación de presentación de resultados

2.5 Problemas encontrados

El desconocimiento previo del funcionamiento de las peticiones a OpenStreetMap ha supuesto problemas a la hora de obtener la información geográfica. Las primeras pruebas se realizaron con la herramienta interactiva Overpass-turbo[26], la cual consume directamente de OpenStreetMap sin imponer restricciones. El primer problema en aparecer fue la imposición de límites de descarga de OpenStreetMap: 50000 nodos por descarga como máximo. Esto se solventó al descubrir la Extended API que se ofrece para grandes volúmenes de datos, pero en la respuesta no se incluía toda la información necesaria, sino los identificadores de nodo y sus relaciones. Este segundo problema se solventó al combinar ambas interfaces, este proceso de descarga se puede ver mejor detallado en Anexo II.

[26] https://overpass-turbo.eu/

La respuesta de la mencionada Extended API se encuentra en formato *xml*, y al recorrer los elementos de una cuadrícula de hasta 0.2x0.2 grados no se encuentran problemas. Es a partir de una cuadrícula de 0.2x0.2 grados según las pruebas realizadas, que no es posible cargar en memoria todo el fichero porque esta desborda. En lugar de utilizar DOM API[27] para Java, interfaz ampliamente utilizada, se cambió a SAX API[28], la cual carga en memoria fragmentos y los recorre evitando copar la memoria.

Tras investigar servicios para delegar la tarea de alineación entre emociones y segmentos, se optó por utilizar las características geográficas de MongoDB. Sin embargo, la herramienta para identificar qué segmento es el más cercano dado un par de coordenadas no era del todo acertada, esta únicamente tiene en cuenta los extremos de los segmentos. Es por ello por lo que se diseñó un algoritmo apoyado en geometría plana para llevar a cabo dicha vinculación, esto se detalla en Anexo III.

El algoritmo al que se hace mención en el anterior problema fue implementado primero en lenguaje Python dada la velocidad y facilidad que suponía su ejecución. Cuando llegó el momento de integrar los diferentes componentes construidos, la integración de este con el resto del sistema se estimó demasiado costosa en tiempo dado que no se tenía experiencia previa en la integración Java-Python, luego se optó por traducir el algoritmo a lenguaje Java.

[27] https://docs.oracle.com/javase/tutorial/jaxp/dom/readingXML.html
[28] https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html

LECCIONES APRENDIDAS Y CONCLUSIONES

3.1 Conocimientos adquiridos

En este proyecto, todos los componentes del sistema han incluido en algún momento un periodo de formación por aplicar tecnologías desconocidas o herramientas relacionadas con la información georreferenciada. El carácter diferencial es el tratamiento de los datos geográficos. Desde su descarga de diferentes API de Open Street Map, con su posterior tratamiento para descartar información irrelevante, hasta su asociación con emociones y el volcado en la base de datos.

La capa lógica de la aplicación de gestión expone una interfaz REST, este concepto ya se ha implementado en proyectos anteriores, pero no con *Spring Boot*. El uso de la biblioteca Quartz de Java, para la programación de tareas de forma automática, también ha supuesto comprender su funcionamiento y configuración.

Exceptuando una sola asignatura realizada al mismo tiempo que este trabajo, nunca se había trabajado con bases de datos no relacionales. Esto ha conllevado un periodo de formación e investigación en el funcionamiento de MongoDB en cuanto a características básicas y características geográficas.

Finalmente, para la aplicación web de presentación de resultados, tanto Angular como la biblioteca Leaflet no habían sido utilizados previamente. En la formación con Angular se incluye la creación de componentes y servicios, en la de Leaflet se incluye la presentación de los resultados sobre un mapa.

3.2 Ideas Futuras

Tal y como está diseñado el sistema, únicamente se tiene información sobre los segmentos que tienen emociones asociadas y, por tanto, a la hora de extraer conocimiento podrían quedar huecos en calles que tuvieran tramos sin vinculaciones. En un futuro se podría plantear un algoritmo de interpolación que asignara a estos segmentos emociones en función de sus adyacentes.

Estrechamente relacionado a dicha interpolación, se podría plantear una evolución del algoritmo de vinculación para las situaciones de conflicto en intersecciones. Para comprobar que los resultados que se obtienen son fiables se ampliaría el proyecto con algún componente externo encargado de asegurar que los datos que se están integrando son veraces.

Como se detalla en Anexo II, la base cartográfica de OpenStreetMap crece con el tiempo, nuevas edificaciones o ajustes de precisión son posibles. El refresco de datos geográficos hace que las emociones vinculadas en ese espacio actualizado pasen por el proceso de búsqueda de segmento más cercano de nuevo. No se tiene en cuenta que demasiadas modificaciones en la realidad, y por tanto en OpenStreetMap, dejen de provocar cierto sentimiento. Se podría plantear una línea de mejora tal que, pasado cierto número de modificaciones o modificaciones de precisión bruscas, las emociones perdieran su valor.

Otra línea de mejora sería la ampliación de las capacidades de filtrado en la presentación de los datos sobre el mapa. Tal y como se ha indicado, los costes de procesado de las consultas hacen inviable una búsqueda libre. Sin embargo, el actual número de consultas podría incrementarse con la generación de nuevas precalculadas.

Finalmente, una posible mejora sobre el sistema actual sería la posibilidad de exportación de datos para su uso en otros contextos. Sobre las preguntas precalculadas, y acotando geográficamente por la zona de visualización que ofreciera el mapa, se podrían exportar los segmentos que aparecieran con la etiqueta de emoción. Estas exportaciones podrían llevarse a cabo sobre formatos geográficos más clásicos (GeoJSON, Shapefile[29], KML[30], etc) o sobre sistemas más modernos orientados a procesamientos en entornos móviles como Geopackage[31].

3.3 Conclusiones

Resultados

El objetivo de este trabajo fin de grado se fijó en realizar un sistema de recolección de emociones en almacenes externos para poder asociarlas a elementos geográficos y con ello servir de base a futuros estudios o sistemas de información, y el mismo ha sido cumplido. El sistema, de forma autónoma, recoge emociones e información geográfica, la vincula y mantiene la base cartográfica actualizada.

Se planteaba que cada zona en la que se desplegara el sistema de recolección de emociones (cabe recordar que este sistema forma parte de otro TFG en desarrollo al mismo tiempo) tuviera su propio almacén, y que el sistema central pudiera añadirlos sin afectar al funcionamiento del resto. Es por ello por lo que se ha dotado a este último de las necesarias herramientas para poder dar de alta los diferentes almacenes externos de forma que podrán ir incorporándose progresivamente.

Al principio del planteamiento no existía una forma estudiada de consumo de datos geográficos. El punto de partida era el trabajo con OpenStreetMap por ser de libre acceso. A partir de aquí ha sido necesario evaluar los costes de descarga de la información de cara a proponer una solución que resulte viable técnica y operacionalmente.

También surgió la necesidad de dar solución al manejo de un gran volumen de datos de cara a la exportación de estos. La generación de consultas precalculadas que se van actualizando con la inclusión de nuevas emociones ha sido la opción elegida ya que se considera que es la única operacionalmente viable para poder contar con tiempos de respuesta en la visualización asumibles por un usuario.

[29] https://es.wikipedia.org/wiki/Shapefile

[30] https://es.wikipedia.org/wiki/KML

[31] https://www.geopackage.org/

Conclusión personal

A nivel personal, este proyecto me ha hecho darme cuenta de lo capaz que puedo ser. Cuando se me planteó por primera vez la hoja de ruta, sabía que se dibujaba un trabajo muy diferente a todo lo realizado hasta el momento. He podido ampliar mis conocimientos en diversas tecnologías que en un futuro inmediato me pueden servir como escaparate profesional.

BIBLIOGRAFÍA

[1] [Kaya et al, 2004] Kaya N., Epps H.H. Relationship between color and emotion: a study of college students. En la College Student Journal, Volumen 38, páginas 396-405, (Estados Unidos, Septiembre, 12-09-2004)

[2] https://elsindromedelahojaenblanco.wordpress.com/2012/11/19/colores-y-emociones/

[3] http://www.ugr.es/~setchift/docs/cualia/sinestesia_colores_emociones.pdf

ANEXO I. FORMACIÓN EN OPENSTREEETMAP

OpenStreetMap (también conocido como OSM) es un proyecto colaborativo para crear mapas editables y libres. En lugar del mapa en sí, los datos generados por el proyecto se consideran su salida principal.

Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, ortofotografías[32] y otras fuentes libres. Esta cartografía, tanto las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuye bajo licencia abierta Licencia Abierta de Bases de Datos[33] (en inglés ODbL).

Los usuarios registrados pueden subir sus trazas desde el GPS y crear y corregir datos vectoriales mediante herramientas de edición creadas por la comunidad OpenStreetMap. Cada semana se añaden 90.000 km de nuevas carreteras con un total de casi 24.000.000 km de viales, eso sin contar otros tipos de datos (pistas, caminos, puntos de interés, etc.). El tamaño de la base de datos (llamada planet.osm) se situaba en julio de 2017 por encima de los 800 gigabytes (58 GB con compresión bzip2).

OpenStreetMap utiliza una estructura de datos topológica[34]. Los datos se almacenan en el datum[35] WGS84 lat/lon (EPSG:4326) de proyección de Mercator[36]. Los datos primitivos o elementos básicos de la cartografía de OSM son:

- Los nodos (nodes). Son puntos que recogen una posición geográfica dada.
- Las vías (ways). Son una lista ordenada de nodos que representa una polilínea o un polígono (cuando una polilínea empieza y finaliza en el mismo punto).
- Las relaciones (relations). Son grupos de nodos, vías u otras relaciones a las que se pueden asignar determinadas propiedades comunes. Por ejemplo, todas aquellas vías que forman parte del Camino de Santiago.
- Las etiquetas (tags). Se pueden asignar a nodos, caminos o relaciones y constan de una clave (key) y de un valor (value). Por ejemplo: highway=trunk define una vía como carretera troncal.

Los atributos de los datos siguen un modelo más elaborado que las folksonomías[37] de indexación social. La ontología de las características del mapa (principalmente el significado de las etiquetas) se mantiene mediante una wiki.

El proyecto facilita varias API destinadas a la edición y ampliación de su base de datos, todas ellas se descartan por existir una exclusivamente para el consumo de información. No obstante, esta tiene impuesto el límite de dos peticiones cada 5 minutos y 50000 nodos por cada una de ellas, luego ha sido necesario recurrir a servicios externos que se apoyan en la misma fuente de datos.

---

[32] https://es.wikipedia.org/wiki/Ortofotograf%C3%ADa
[33] https://es.wikipedia.org/wiki/Licencia_Abierta_de_Bases_de_Datos
[34] https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n_geogr%C3%A1fica#Modelo_topol.C3.B3gico
[35] https://es.wikipedia.org/wiki/Datum
[36] https://es.wikipedia.org/wiki/Proyecci%C3%B3n_de_Mercator
[37] https://es.wikipedia.org/wiki/Folcsonom%C3%ADa

Overpass XAPI[38] es una interfaz de solo lectura de la cual se extrae información por medio de peticiones *http*. Esta no impone límites de descarga, pero no devuelve toda la información referida a cada nodo o vía, se precisa de combinar dicho servicio con la API general de OpenStreetMap (ver Anexo II).

[38] https://wiki.openstreetmap.org/wiki/Overpass_API/XAPI_Compatibility_Layer

ANEXO II. DESCARGA DE DATOS GEOGRÁFICOS

Primera descarga de los datos

Se tiene en cuenta que las coordenadas asociadas a una emoción pueden situarse en cualquier posición del globo terráqueo, pero no es asumible descargar toda la información de OpenStreetMap. No tendría sentido por su coste en tiempo y espacio de almacenamiento, para luego no usarlos nunca. Es por ello por lo que se ha dividido el mundo en una cuadrícula global de 0.2 de ancho y 0.2 de alto. Es decir, se ha dividido el mundo en una matriz de 900 x 450, cuya codificación va de (-180) a (+180) en el eje X y de (-90) a (+90) en el eje Y. La decisión del tamaño de cuadrícula se detalla posteriormente.

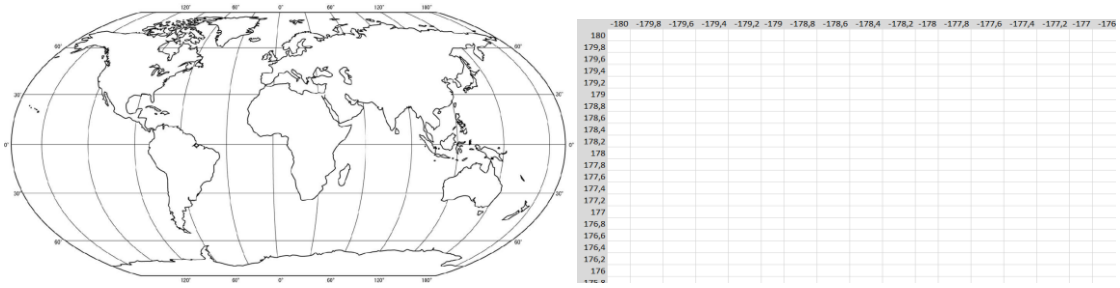

Figura 13. Matriz de cobertura mundial

Cuando se toma un punto al que está vinculado una emoción para alinearla a un segmento de calle, se verifica si hay datos de la correspondiente casilla en la base de datos, y si no se descargan tal y como se muestra en la siguiente figura.

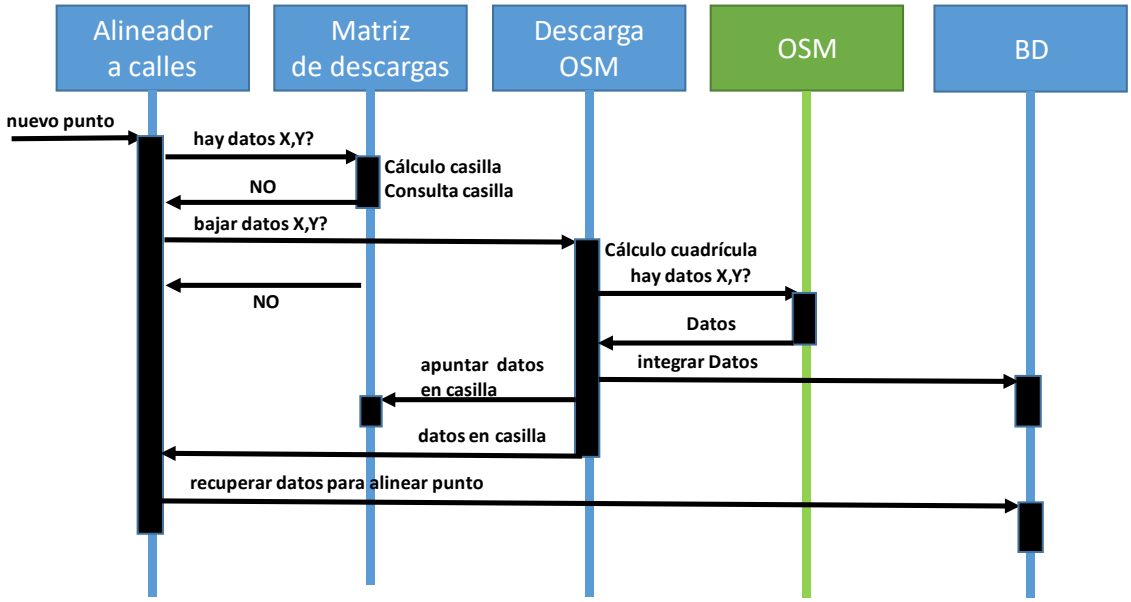

Figura 14. Flujo de verificación y descarga de datos

Como se detalla en el Anexo I, es necesaria la combinación de diferentes API para la descarga de los datos de OpenStreetMap. En primer lugar, se realiza una petición a Overpass XAPI[39] incluyendo las coordenadas de la cuadrícula a descargar, la respuesta obtenida contiene una

[39] Ejemplo de petición: http://www.overpass-api.de/api/xapi?way[bbox=-1,41.6,-0.8,41.8]

lista con todos los elementos dentro del cuadrilátero definido. De dicha lista se extraen los identificadores de todas las vías, estas están marcadas con la etiqueta *way*, y se procede a consultar una a una su información detallada con la interfaz general de OpenStreetMap[40]. De cada respuesta se extraen los nodos que componen la vía, con ello se puede almacenar en la base de datos cada segmento conformado por un par de nodos consecutivos y sus coordenadas. Una vez finalizado el volcado de todos los segmentos de calle, la casilla se marca como descargada.

El proceso de consulta de casilla que se opera en la matriz de descargas tiene en cuenta las coordenadas propias del punto, y el posible error en la precisión del punto ofrecido por el GNSS[41] (zona en amarillo en la figura siguiente). Es por ello por lo que maneja un buffer tal que, si las coordenadas caen en el mismo, no solamente se verifica una casilla, sino también la siguiente. De esta manera, no solamente se cubren los posibles errores de precisión del sistema de localización, sino que también se garantiza que en el alineamiento de segmentos se está teniendo en cuenta segmentos que quedan fuera de la casilla, pero podrían ser candidatos. En el ejemplo, el segmento más al Este del punto de la calle Tejera podía no haber estado incluido y es candidato al alineamiento.

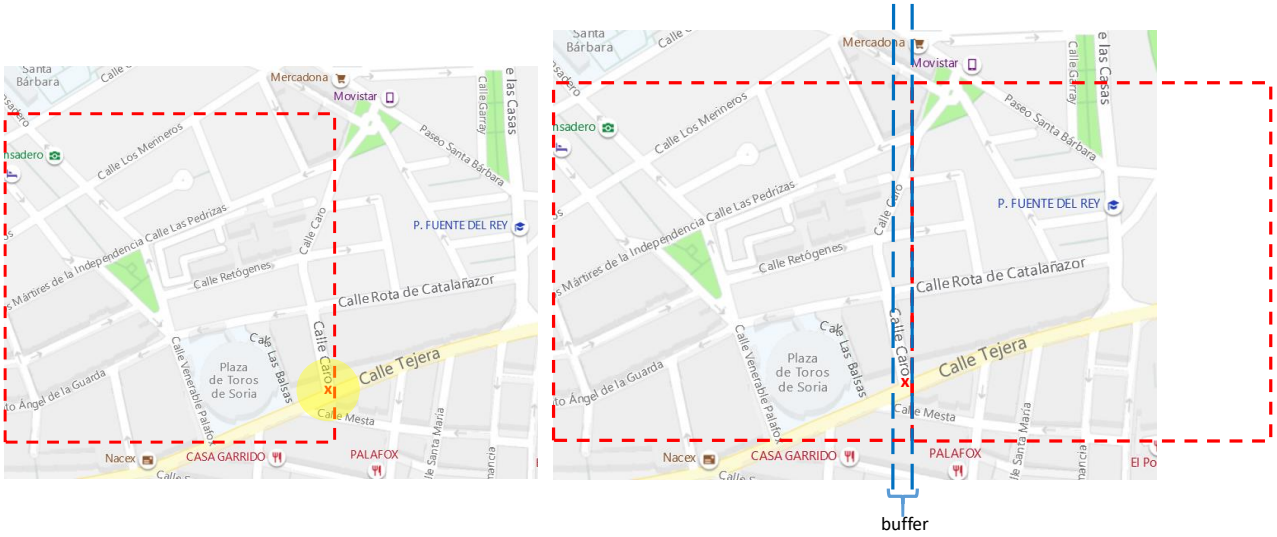

Figura 15. Zonas límite de las casillas

La determinación sobre si es una o más cuadrículas se establece en función de si el punto cae en la zona de no solape entre casillas (punto 1 en la figura), en el 2.5 % del borde exterior de la cuadrícula entre dos cuadrículas (punto 2 en la figura) o en la zona de solape de cuatro cuadrículas (punto 3 en la figura).

[40] Ejemplo de petición: https://api.openstreetmap.org/api/0.6/way/209223874/full.json
[41] https://es.wikipedia.org/wiki/Sistema_global_de_navegación_por_satélite

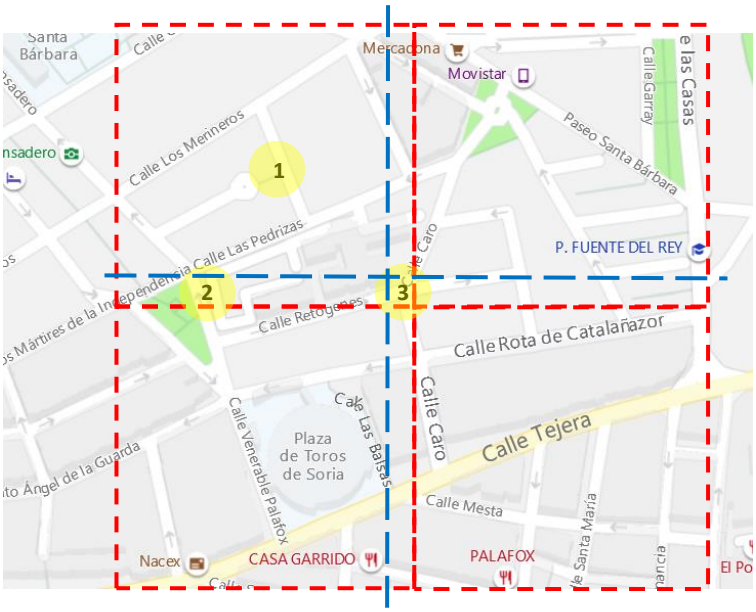

Figura 16. Determinación de casillas a descargar

La determinación del tamaño de la casilla se ha llevado a cabo analizando las necesidades de cómputo que van asociadas al proceso de descarga. Inicialmente se planteó usar casillas de 0.5 x 0.5. En el siguiente ejemplo se puede observar un área de esta dimensión que cubre todo Madrid y hasta más al sur de Aranjuez (40.478000, -3.845000; 39.978000, -3.345000). En esta zona (el recuadro en rojo con línea discontinua)[42], la consulta a OpenStreetMap nos devuelve un total de 443152 calles. El proceso de descarga de datos y volcado en la base de datos necesita de 13,26 horas.

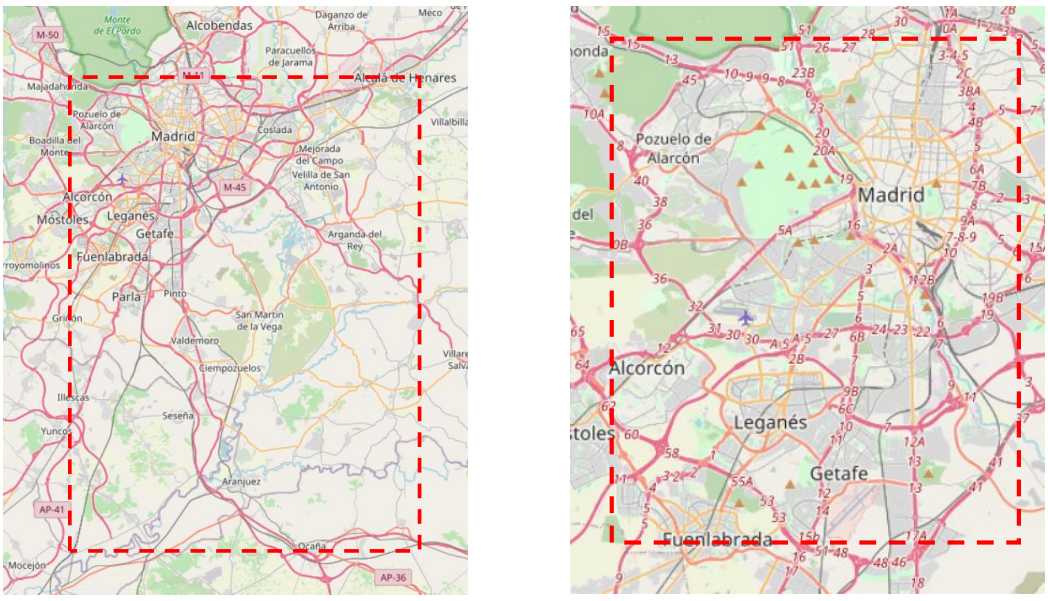

Figura 17. Cuadrículas de 0.5 x 0.5 (izquierda), y 0.2 x 0.2 (derecha)

[42] El hecho de que parezca alargado y no de la sensación de cuadrado se debe a la proyección de la imagen de la Tierra (más o menos esférica) sobre un plano. Ver https://es.wikipedia.org/wiki/Proyección_cilíndrica

Si se reduce la cuadrícula a 0.2 x 0.2 se sigue teniendo todo el centro de Madrid y gran parte de los municipios de su extrarradio (40.478000, -3.845000; 40.278000, -3.645000), lo que se obtiene un total de 244325 calles. El proceso de descarga de datos y volcado en la base de datos necesita de 7,31 horas.

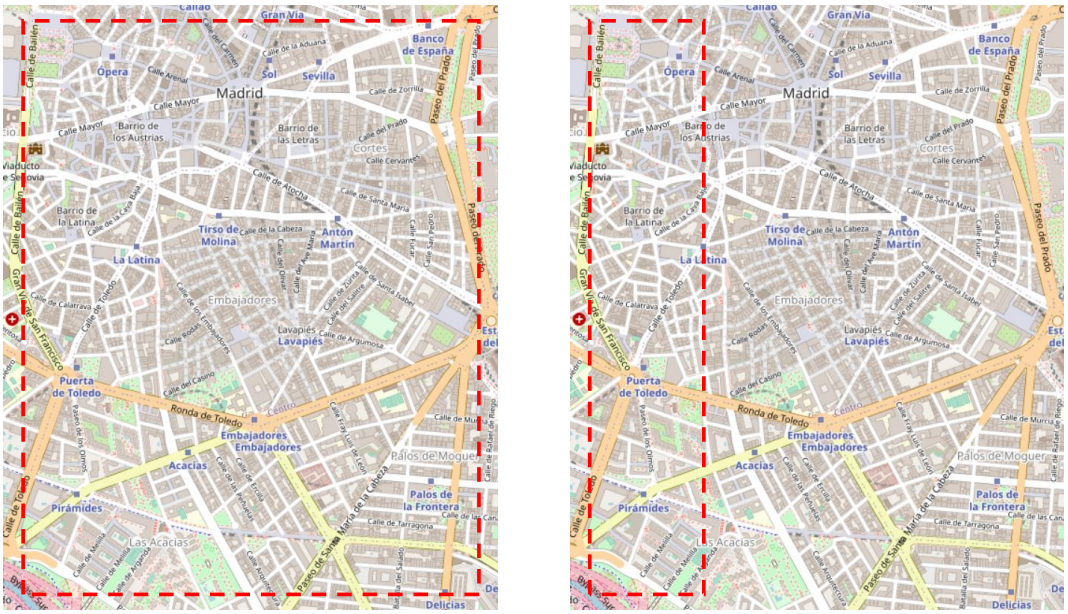

Figura 18. Tamaño de las zonas de solape al 10 % (izquierda), y al 2.5 % (derecha)

Para el cálculo del tamaño de las zonas de solape que debería llevar a la carga de una nueva cuadrícula, se planteó inicialmente usar un 10 % del ancho de la celda. Si se usa el 10 % de 0.2, se tiene que habría una franja de 0.02 x 0.2 en los laterales, y de 0.2 x 0.02 en la parte de arriba y de debajo de la cuadrícula. A modo de experimento, se ha tomado una cuadrícula de 0.02 x 0.02 en el centro de Madrid (se ha buscado un escenario con una alta densidad de calles) cubriendo entre Plaza de Oriente y la estación de Atocha (40.419600, -3.713000; 40.399600, -3.693000). En esta zona, la consulta a OpenStreetMap nos devuelve un total de 9184 calles y 84117 nodos. Esto lleva a considerar ese 10 % como un tamaño demasiado grande, por lo que se ha optado por reducirlo a un 2.5 %. En este caso se ha repetido el experimento en la zona centro de Madrid con una franja de 0.005 x 0.02 (40.419600, -3.713000; 40.399600, -3.708000) y se ha obtenido de OpenStreetMap un total de 2395 calles y 21563 nodos.

A continuación, se expone el código fuente encargado de indicar qué casillas deben ser descargadas para un par de coordenadas.

```
/**
     * Returns an ArrayList with the quadrant where the position belongs, and its
     * adjacent ones if they are close enough
     *
     * @param p             Position of the point we do not know where belongs
     * @param intervalSize  One side size, in grades, of the matrix quadrant (e.g.
     *                      0.2)
     * @param intervalLimit Matrix quadrant portion used to define when the Position
     *                      is too closed to an edge (e.g. 0.1)
     *
     * @return e.g. in getMatrixQuadrantList(p, 0.2, 0.1) the quadrant size is
     *         0.2x0.2 grades and the height of the portion where the Position is
     *         too close to another quadrant is 10% of 0.2
     */
```

```
    public static ArrayList<MatrixQuadrant> getMatrixQuadrantList(Position p, double inter
valSize, double intervalLimit) {
        ArrayList<MatrixQuadrant> list = new ArrayList<MatrixQuadrant>();

        double top, bottom, left, right;

        if (p.getLat() >= 0) {
            bottom = truncate(p.getLat() / intervalSize) * intervalSize;
        } else if (p.getLat() / intervalSize - truncate(p.getLat() / intervalSize) == 0) {
            bottom = truncate(p.getLat() / intervalSize) * intervalSize;
        } else {
            bottom = truncate(p.getLat() / intervalSize - 1) * intervalSize;
        }
        top = bottom + intervalSize;

        if (p.getLng() >= 0) {
            left = truncate(p.getLng() / intervalSize) * intervalSize;
        } else if (p.getLng() / intervalSize - truncate(p.getLng() / intervalSize) == 0) {
            left = truncate(p.getLng() / intervalSize) * intervalSize;
        } else {
            left = truncate(p.getLng() / intervalSize - 1) * intervalSize;
        }
        right = left + intervalSize;

        // Main Quadrant
        MatrixQuadrant m1 = new MatrixQuadrant(bottom, top, left, right,null);
        list.add(m1);

        // Here it is checked if the position is too close to any edge of the quadrant
        boolean nearTop = false, nearBottom = false, nearLeft = false, nearRight = false;
        if (p.getLat() <= bottom + intervalSize * intervalLimit) {
            nearBottom = true;
        } else if (p.getLat() >= top - intervalSize * intervalLimit) {
            nearTop = true;
        }

        if (p.getLng() <= left + intervalSize * intervalLimit) {
            nearLeft = true;
        } else if (p.getLng() >= right - intervalSize * intervalLimit) {
            nearRight = true;
        }

        // Non diagonal coincidences
        if (nearBottom) {
            MatrixQuadrant m2 = new MatrixQuadrant(bottom - intervalSize, top - intervalSi
ze, left, right, null);
            list.add(m2);
        } else if (nearTop) {
            MatrixQuadrant m2 = new MatrixQuadrant(bottom + intervalSize, top + intervalSi
ze, left, right, null);
            list.add(m2);
        }
        if (nearLeft) {
            MatrixQuadrant m2 = new MatrixQuadrant(bottom, top, left - intervalSize, right
 - intervalSize, null);
            list.add(m2);
        } else if (nearRight) {
```

```
            MatrixQuadrant m2 = new MatrixQuadrant(bottom, top, left + intervalSize, right
 + intervalSize, null);
            list.add(m2);
        }

        // Diagonal coincidences
        if (nearBottom && nearLeft) {
            MatrixQuadrant m4 = new MatrixQuadrant(bottom - intervalSize, top - intervalSi
ze, left - intervalSize, right - intervalSize, null);
            list.add(m4);
        } else if (nearBottom && nearRight) {
            MatrixQuadrant m4 = new MatrixQuadrant(bottom - intervalSize, top - intervalSi
ze, left + intervalSize, right + intervalSize, null);
            list.add(m4);
        } else if (nearTop && nearRight) {
            MatrixQuadrant m4 = new MatrixQuadrant(bottom + intervalSize, top + intervalSi
ze, left + intervalSize, right + intervalSize, null);
            list.add(m4);
        } else if (nearTop && nearLeft) {
            MatrixQuadrant m4 = new MatrixQuadrant(bottom + intervalSize, top + intervalSi
ze, left - intervalSize, right - intervalSize, null);
            list.add(m4);
        }
        return list;
    }

/* Returns the non-decimal part of the given double */
public static int truncate(double x) {
            return (int) x;
}
```

Actualización y refresco de datos

Una de las principales características de OpenStreetMap es que es una base de datos viva que va evolucionando con el tiempo. Esta evolución viene de dos razones principales. Por una parte, los propios cambios en la realidad son llevados al sistema por los editores. Por otro lado, el sistema va creciendo por refinamientos sucesivos que aportan más nivel de detalle. Una aproximación clásica al mapeo de alguna zona (tanto rural como urbana) es reflejar inicialmente las vías principales para, en posteriores trabajos de edición, añadir vías de menor entidad.

Esta dinamicidad de los datos tiene que ser trasladada al sistema de emparejamientos que se desarrolla en este trabajo fin de grado. Para ello se ha creado el subsistema de actualización de datos que se menciona en la memoria. Este subsistema se apoya en la matriz de casillas atendiendo al siguiente comportamiento. Cuando el sistema se despliega inicialmente, no existe ninguna casilla. Cada vez que se descargan los datos de una de ellas, la casilla se crea y se etiqueta con el sello de tiempo del proceso de descarga. De este modo, cuando el subsistema de actualización entra a trabajar, busca aquellas casillas más antiguas y procede a ejecutarse sobre ellas. Para la primera implementación llevada a cabo, las casillas más antiguas serán las que se descargaron por última vez hace más de 6 meses. No obstante, éste es un parámetro configurable que se podría ajustar si se identifican problemas porque los datos se refresquen con demasiada o demasiada poca frecuencia.

Implementación de la matriz de casillas

La matriz no es excesivamente grande, por lo que resulta fácil de recorrer sea cual sea la implementación que la soporte. Lo que resulta más crítico es garantizar la persistencia de la información que tiene almacenada dado que supone una visión de los datos de OpenStreetMap que se tienen en la base de datos. Tampoco sería muy problemático si se perdiera esta información dado que podría reconstruirse a partir de los puntos con las emociones. Con todos estos condicionantes, finalmente se ha optado por dar soporte a la implementación de la matriz mediante una colección de casillas y otra de segmentos, ambas dos en el primer arranque se encuentran vacías y se van ampliando con cada descarga. Los segmentos están conformados por el identificador de la calle a la que pertenecen, los identificadores de los dos nodos que definen el segmento, las coordenadas de cada uno de ellos y la fecha y hora de su última descarga. Por otro lado, cada casilla se define por las coordenadas de sus bordes luego no es posible la descarga de casillas por duplicado, además incluyen el sello temporal de su última descarga.

ANEXO III. VINCULACIÓN DE EMOCIONES Y SEGMENTOS

Herramientas candidatas

Es necesario poder relacionar las coordenadas que llegan como entrada al sistema con los tramos de calle para que nos aporten valor. Este proceso es llamado geocodificación inversa[43], y se han barajado como opción varios servicios para delegar el trabajo en ellos.

La primera es el uso de Google Maps[44], se descarta totalmente por dos motivos principales: no es gratuito y nos devolvería resultados difíciles de tratar. La base de datos extraída de OpenStreetMap representa tramos de calle sin importar el número de portal sino las intersecciones entre calles, en cambio el servicio de Google haría lidiar con problemas que no aportarían beneficios. Probablemente se tendría que almacenar más información de la necesaria. Otra alternativa es el uso de LocationIQ[45] que, aunque gratuito, presenta el mismo problema que Google Maps en cuanto al tratamiento y almacenamiento de datos.

Finalmente, se decide descartar servicios externos y emplear el módulo geoespacial de MongoDB[46] por no suponer grandes problemas en el tratamiento de la información geográfica, además de ningún coste por su uso.

Primeras pruebas con volumen de datos reducido

Para las primeras pruebas de vinculación entre segmentos y pares de coordenadas, gracias a scripts en Python, se ha generado en un fichero JSON un conjunto reducido de segmentos de calle contenidos en una cuadrícula de 20x20 unidades. Dichos segmentos se han importado[47] en MongoDB y se ha creado un índice 2d[48] para datos en planos de dos dimensiones. Al realizar las consultas esperando obtener qué segmento es más próximo dado un par de coordenadas, se descubre que realmente el obtenido es aquel que contiene el nodo más cercano.

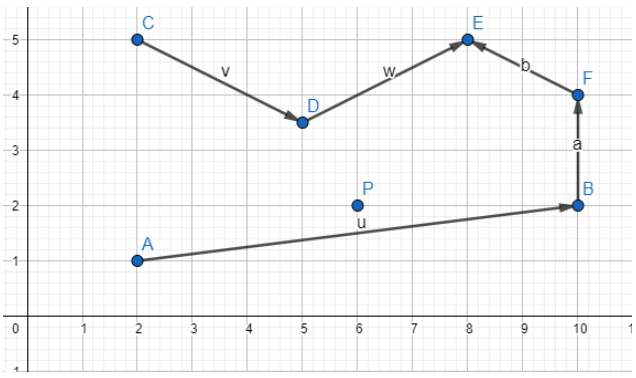

Figura 19. Prueba errónea de vinculación de emociones y segmentos

[43] https://es.wikipedia.org/wiki/Geocodificaci%C3%B3n_inversa
[44] https://cloud.google.com/maps-platform?hl=es
[45] https://locationiq.com
[46] https://docs.mongodb.com/manual/geospatial-queries/
[47] https://docs.mongodb.com/manual/reference/program/mongoimport/
[48] https://docs.mongodb.com/manual/core/2d/

Este resultado no es válido para todos los casos como se puede ver en la figura anterior, el punto P debería asociarse al segmento AB pero, como el vértice D es más próximo, los segmentos CD y DE se identifican como más cercanos a P.

MongoDB no incorpora ninguna herramienta para calcular lo descrito anteriormente de forma correcta, luego es necesario comprobar por geometría plana si el resultado devuelto es correcto. En una segunda prueba se plantea en vez de obtener un segmento, obtener los tres más cercanos, y como se indica en la figura comprobar a cuál pertenece realmente.

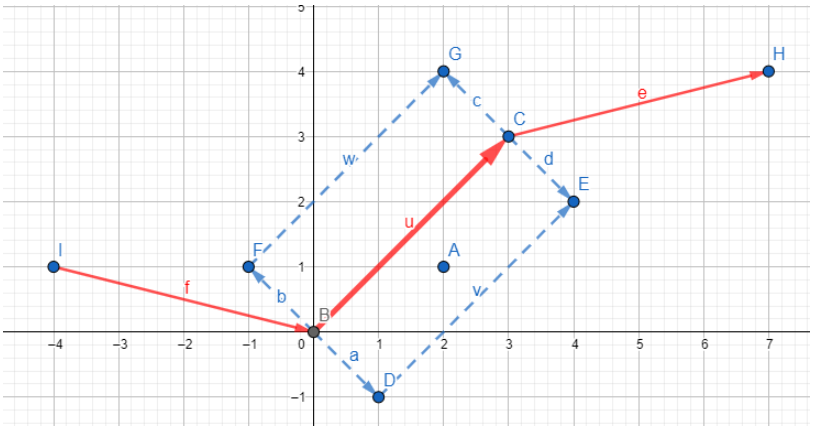

Figura 20. Cálculo de segmento más cercano por geometría plana

En la figura anterior se representan los tres segmentos más cercanos al punto *A* devueltos por la base de datos. Para cada segmento se calcula su rectángulo asociado, en el caso del ejemplo el punto *A* entra dentro del área del segmento *u*.

Para comprobar si el punto A se encuentra dentro del cuadrilátero formado por los puntos D, E, F y G, se comprueba que el área de dicho polígono es igual a la suma de las áreas de los triángulos definidos por el vértice A y cada par de vértices consecutivos del cuadrilátero: área(D,E,F,G) = área(A,D,E) + área(A,E,F) + área(A,F,G) + área(A,G,D).

Pruebas con datos geográficos

Previo a la realización de las pruebas ha sido necesaria la instalación de QGIS[49] para la visualización sobre el mapa de los segmentos y puntos geográficos. También ha sido necesario cambiar el índice 2d mencionado en las primeras pruebas por un índice 2dsphere[50] para geometrías esféricas. Además, se ha definido un script en Python para transformar la información de los segmentos obtenida de OpenStreetMap en un fichero de extensión GeoJSON[51] que pueda ser importado en QGIS.

[49] https://www.qgis.org/es/site/
[50] https://docs.mongodb.com/manual/core/2dsphere/
[51] https://es.wikipedia.org/wiki/GeoJSON

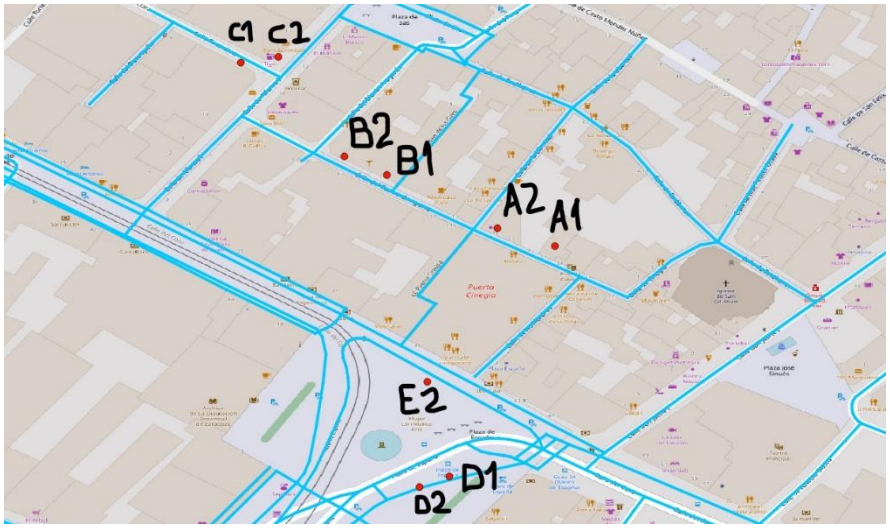

Figura 21. Prueba de vinculación de emociones y segmentos con QGIS Desktop

Para las pruebas con datos geográficos se ha adaptado el algoritmo de vinculación a segmentos. En este caso se cuenta con dos pares de coordenadas que nos ayudan a definir el movimiento del usuario y resolver situaciones de conflicto en la vinculación de segmentos. Además, se ha mejorado la construcción del rectángulo que comprende la cercanía a un segmento. El punto más lejano dentro de dicha cercanía se encontrará a 10 metros del segmento, siendo este valor fácilmente modificable si se viera que fuera demasiado o demasiado poco. Se asume que la curva del planeta no afecta de forma significativa al cálculo de áreas que conforman el algoritmo.

Teniendo en cuenta que un punto puede estar en una intersección entre calles, cabe la posibilidad de que se encuentre contenido dentro de varios cuadriláteros que envuelven diferentes tramos de calle. Las pruebas se han limitado a obtener como máximo dos segmentos cercanos y tratar sus posibles choques con una política de conflictos.

Las colisiones que se pueden dar son los siguientes:

- Ambos pares de coordenadas se encuentran en la misma intersección (se toma como referencia el primer par de coordenadas).
- Ambos pares de coordenadas se encuentran en intersecciones, pero no en la misma (se toma como referencia el primer par de coordenadas).
- Un par de coordenadas se encuentra en una intersección, pero el otro par no (se toma como referencia el par de coordenadas fuera de la intersección).
- Ningún par de coordenadas se encuentra en una intersección, pero el cálculo de su segmento más cercano no resulta el mismo segmento (se toma como referencia el primer par de coordenadas).

Con la finalización exitosa de estas pruebas se asume que el algoritmo no debe recibir más modificaciones.

ANEXO IV. APLICACIÓN WEB DE BÚSQUEDA Y PRESENTACIÓN DE INFORMACIÓN

El problema de la búsqueda y presentación de resultados

Como ya se ha visto a lo largo de la memoria, en el almacenamiento de las emociones se establece una relación de cada una de ellas con el segmento de calle al que se ha vinculado. De este modo, para cada segmento vamos a tener una lista de emociones que tendrán los campos ya descritos. A la hora de establecer el color con el que se va a representar ese segmento de calle en un mapa, "contaremos" el número de emociones de cada tipo y le asignaremos a la calle el color de la que más allá. Esto lo haremos de acuerdo con el siguiente patrón de colores:

- Relajado – blanco
- Contento – verde
- Neutro – azul oscuro
- Asqueado – rojo.
- Nervioso – naranja

La decisión de escoger los colores viene dada según connotaciones obtenidas de artículos [1] y páginas web encontradas [2][3]. El color blanco representa paz, orden, conceptos estrechamente relacionados a la relajación. Para el estado neutro se ha escogido el azul oscuro, es un color de la gama de colores fríos, evoca sensación de frialdad y tranquilidad, según las fuentes citadas se relaciona con la calma. La emoción de felicidad viene representada por el color verde, un color más cálido que el azul. El verde está asociado a la vida y la naturaleza, provocando que una persona esté contenta o entusiasmada. Por otro lado, la emoción de asco se representa mediante el color rojo. La paleta de colores cercana a este se relaciona fuertemente con la amargura, la molestia o la irritabilidad. Por último, la sensación de nerviosismo, precaución y ansiedad se asocian a tonos naranjas. Es un tono que se utiliza ampliamente para situaciones de alerta.

El proceso de búsqueda y presentación es lento dado que debemos tomar cada una de las calles y efectuar los cálculos correspondientes (recordemos que la visualización de una ciudad como Madrid pueden ser cientos de miles de segmentos de calle). En este sentido, es necesario que se haga el procesado en el momento que mayor control de modificación de los datos se ha llevado a cabo. Este momento es cuando se recolectan las emociones de una base de datos externa para incorporarla al sistema. Es entonces cuando se realiza la vinculación de las emociones a los segmentos de calle y, por tanto, la mejor ocasión para efectuar estos cálculos.

Con esta operativa tendremos la visión de todas las calles en todo su conjunto. Ahora es cuando querríamos poder aprovechar la información adicional que está recogiendo el sistema de recolección de emociones para poder filtrar esta visión de conjunto. El filtrado supone la eliminación de emociones en el procesado de la información para etiquetar con un color a un segmento. Por tanto, volvemos a tener el mismo problema de carga que se nos plantea para el sistema en su conjunto. Es por ello por lo que se ha optado por fijar una serie de filtros que llevan asociado el precálculo de las respuestas (mapas de colores en las calles) que se presentarán en el visor geográfico. Los filtros que se ha decidido fijar son:

- Tramos horarios cualitativos (mañana/tarde/noche) que se vincularán a los siguientes tramos horarios: 6 a 13 horas (mañana); 13 a 21 horas (tarde); 21 a 6 horas (noche).

- Épocas del año: primavera, verano, otoño, invierno (se toman como referencia las siguientes fechas para el cambio de estación: 21 de marzo, 21 de junio, 21 de septiembre y 21 de diciembre; además, las estaciones representadas son en función del hemisferio norte).
- Género: M/F/O.
- Edad: 0 - 18; 18 – 65; más de 65.
- Tipo de usuario: turista/ciudadanía.

Se es consciente de que se están reduciendo mucho las posibilidades de filtrado. No obstante, la combinación de todas estas opciones supone la necesidad de generar 216 mapas, además del global. El trabajo de este TFG dejará abierta las puertas a incrementar las posibilidades de filtrado desagregando los niveles ya diseñados, o incluyendo nuevos juegos de restricciones.

Descripción técnica de la aplicación

La aplicación de presentación de resultados está compuesta de una estructura más sencilla que el sistema principal. La capa de visualización se compone de un único componente (*HomeComponent*) y un único servicio (*DatabaseAccessService*). En la capa lógica también se encuentra un único enrutador y controlador. Se incluyen dos modelos de datos por tratar con emociones y segmentos.

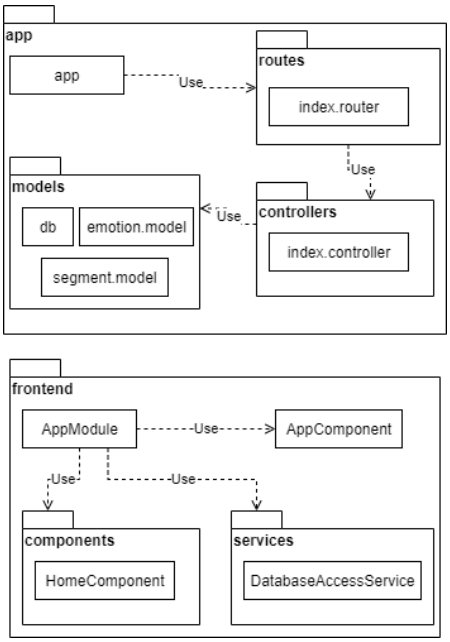

Figura 22. Diagrama de módulos de la aplicación de presentación de resultados

La lógica de la aplicación se construye sobre Node.js y la capa de presentación con Angular siguiendo el patrón MVC[52]. Esta primera, se apoya en la biblioteca Mongoose para pedir a la base de datos los segmentos acordes a los parámetros que recibe por parte del usuario.

[52] https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

El mapa que el usuario visualiza viene dado por la biblioteca Leaflet, que a su vez se configura para consumir de OpenStreetMap. Por lo tanto, existe una concordancia perfecta entre mapa y segmentos dibujados.

Como se ha detallado previamente, se necesita generar 216 mapas precalculados cuando se tiene el mayor control sobre los datos. Esto se realiza creando en MongoDB una colección para cada posibilidad, con lo que serán conjuntos de información completos y accesibles sin necesidad de posterior filtrado.

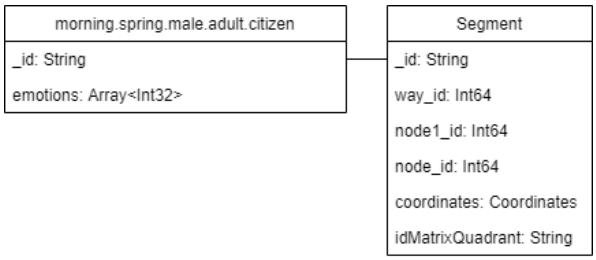

Figura 23. Modelo de datos para las consultas precalculadas

Cuando una emoción ya ha sido asociada a un segmento, tenemos toda la información necesaria para actualizar la información de las colecciones. Se accede a la colección correspondiente, se busca si el segmento ha sido ya incluido, en caso afirmativo se actualiza su vector de emociones, en caso contrario se inicializa.

Cada colección se denomina concatenando los filtros posibles de la siguiente forma: <*tramo horario*>.<*época del año*>.<*género*>.<*rango de edad*>.<*tipo de usuario*>. De esta forma, con los filtros se genera el nombre de la colección y se extraen todos los segmentos que almacena. Cada segmento incluye un vector de enteros de longitud 5 y cada posición indica el número de emociones de cada tipo que tiene asociado según los filtros.

Véase el siguiente ejemplo:

- El usuario selecciona *morning*, *spring*, *male*, *adult* y *citizen*.
- La colección a la que se accede es *morning.spring.male.adult.citizen*.
- Como el campo *_id* se corresponde con el de la entidad *segment*, podemos navegar entre ambas entidades y recuperar las coordenadas para pintarlo en el mapa
- El vector *emotions* es [0, 3, 56, 3, 6]. Por lo tanto sabemos que 0 veces se ha registrado la emoción 1 para este segmento y estos parámetros, 3 veces se ha registrado la emoción 2, 56 veces se ha registrado la emoción 3, …
- Con dicha información podemos pintar el color de la emoción más recurrente.

ANEXO V. MANUAL DE USUARIO

A continuación, se detalla el funcionamiento de las aplicaciones haciendo un recorrido por sus interfaces.

Aplicación de administración

Nada más acceder a la aplicación, se pide al administrador que introduzca sus credenciales.

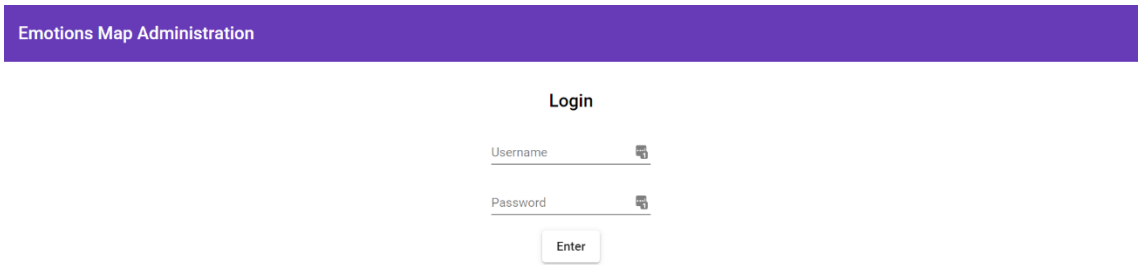

Figura 24. Inicio de sesión

Al pulsar *Enter* el administrador ve la página de gestión de almacenes externos.

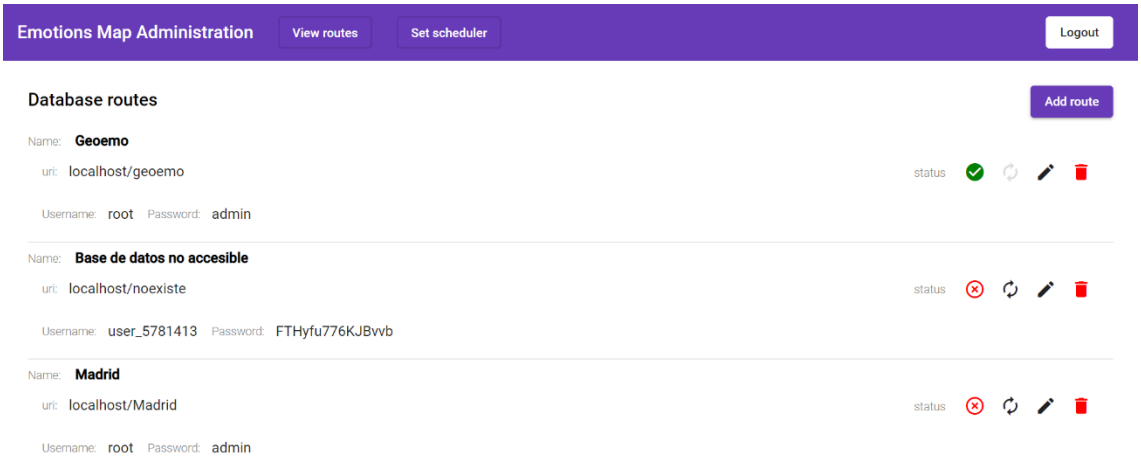

Figura 25. Gestión de almacenes externos

Desde la página principal se puede acceder a la gestión de tareas programadas, añadir un nuevo almacén, editar o eliminar uno existente y comprobar si alguno de los existentes es accesible o no.

Para eliminar un almacén se selecciona el botón con forma de papelera de color rojo, entonces la aplicación preguntará al administrador si está seguro de su decisión.

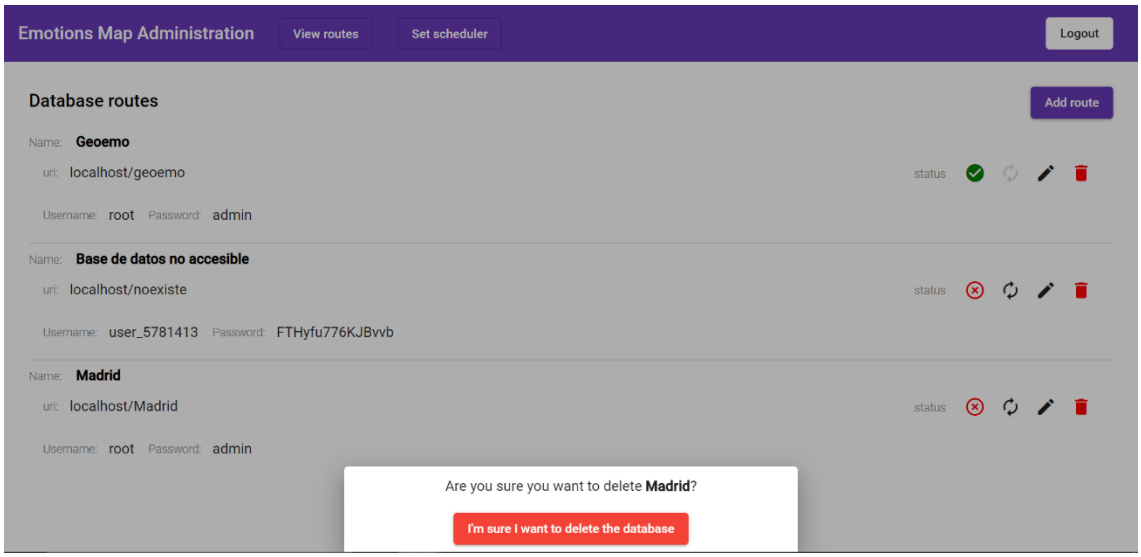

Figura 26. Confirmar eliminación de una ruta

Si el administrador confirma su elección el almacén se elimina de forma definitiva. Tanto en la página para añadir un nuevo almacén como en la de editar se piden las mismas entradas de datos que definen una ruta: nombre, uri y credenciales.

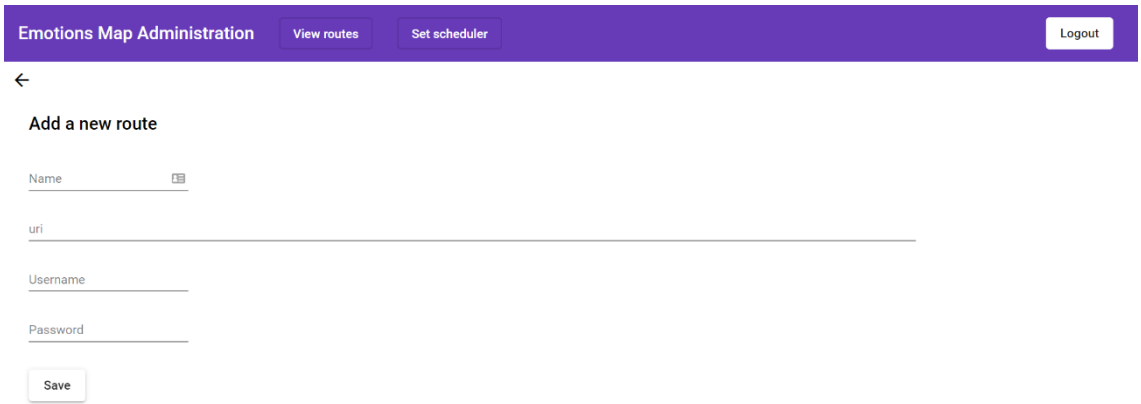

Figura 27. Añadir una nueva ruta

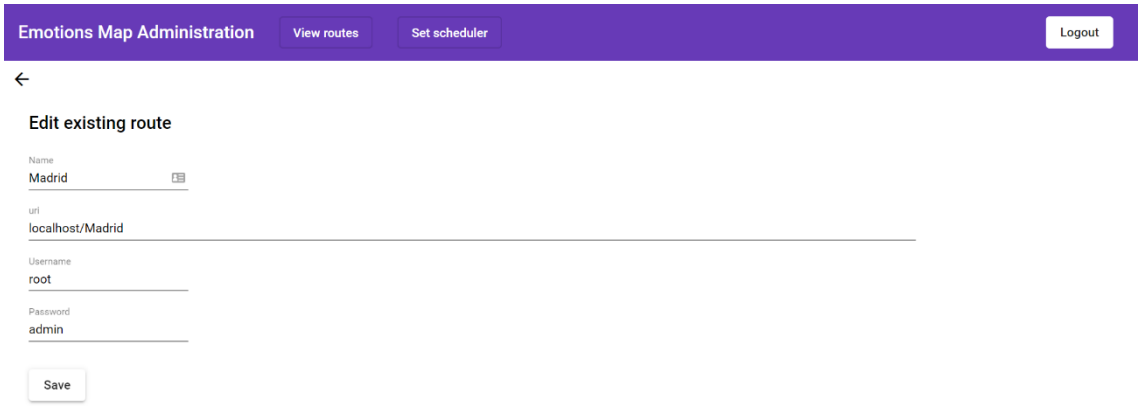

Figura 28. Editar ruta existente

En la parte superior de la aplicación se encuentra el botón *Set Scheduler*, este dirige a la pestaña de configuración de tareas desde la cual el administrador puede añadir, editar, parar, pausar y eliminar tareas.

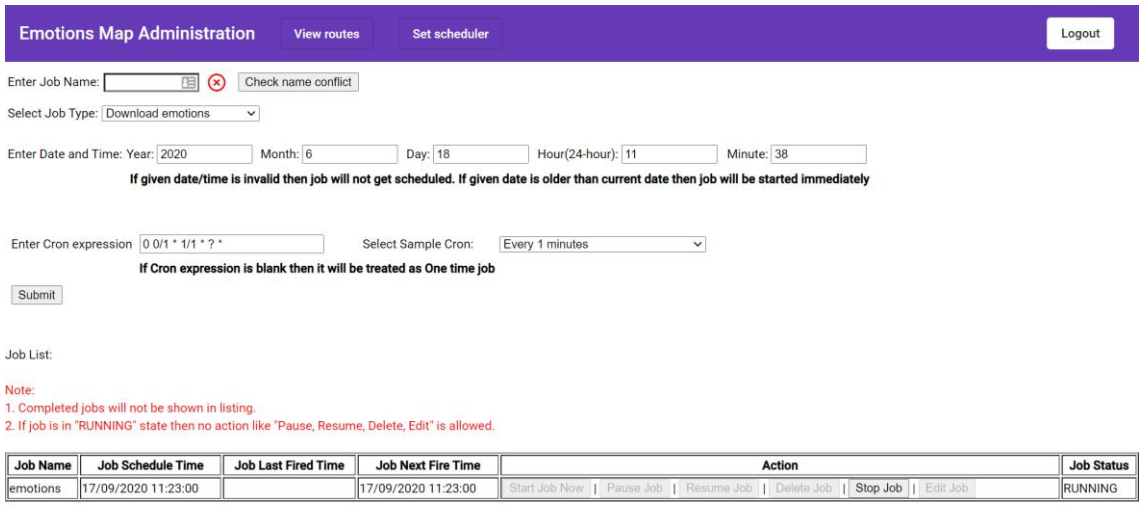

Figura 29. Gestión de tareas programadas

Como se aprecia en la figura anterior, existen indicaciones a modo de aclaración de casos que podrían generar confusión

Si el administrador decide volver a la gestión de rutas puede hacerlo clicando el botón *View routes*. Finalmente, en la parte superior derecha se encuentra el botón *Logout* que cierra la sesión y devuelve al administrador a la primera figura.

Aplicación de presentación de resultados

En esta segunda aplicación, el usuario tiene a su disposición distintos selectores para cambiar el filtro que se aplica a los segmentos. Pulsando sobre el botón *Filter* se cargan los segmentos por colores.

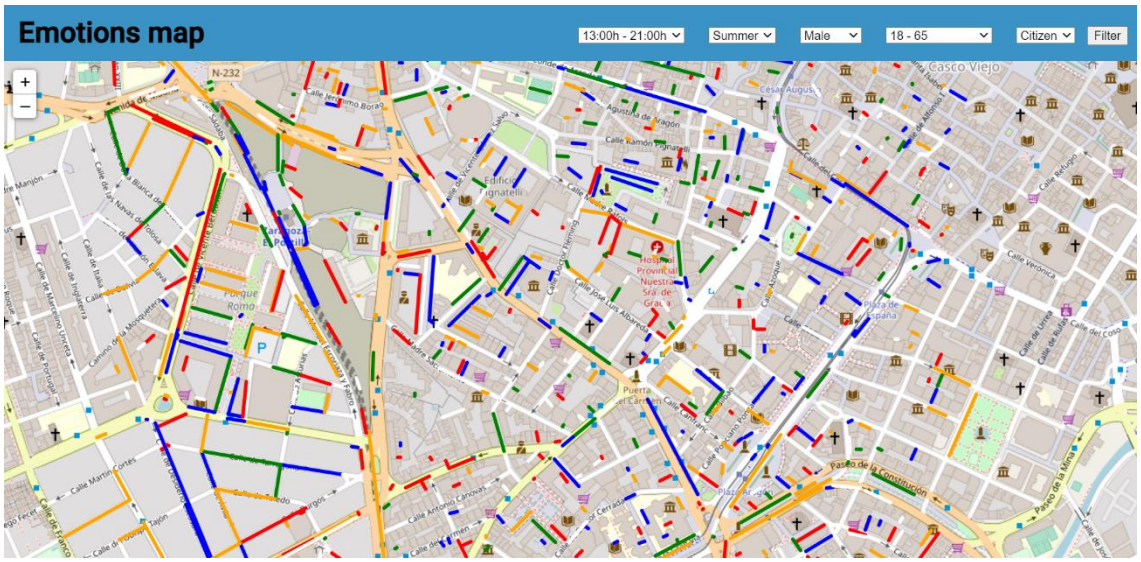

Figura 30. Aplicación de presentación de resultados

Dichos selectores permiten filtrar por:

- Tramos horarios cualitativos (mañana/tarde/noche) que se vincularán a los siguientes tramos horarios: 6 a 13 horas (mañana); 13 a 21 horas (tarde); 21 a 6 horas (noche).
- Épocas del año: primavera, verano, otoño, invierno (se toman como referencia las siguientes fechas para el cambio de estación: 21 de marzo, 21 de junio, 21 de septiembre y 21 de diciembre; además, las estaciones representadas son en función del hemisferio norte).
- Género: M/F/O.
- Edad: 0 - 18; 18 – 65; más de 65.
- Tipo de usuario: turista/ciudadanía.

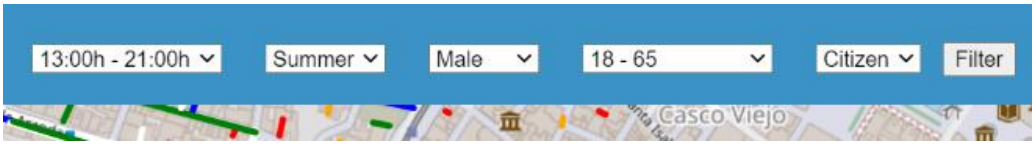

Figura 31. Filtros disponibles en la aplicación de presentación de resultados